Tobias Wantzen

MS Word als Textlieferant für DTP-Programme

Tobias Wantzen

MS Word als Textlieferant für DTP-Programme

Regel Nr. 2: »Machen Sie den bestmöglichen Zug.«
Arthur Bisguier, Schachgroßmeister [17]

Tobias Wantzen

MS Word als Textlieferant für DTP-Programme

Diplomarbeit im Studiengang
Mediapublishing und Verlagswirtschaft

Vorgelegt von Tobias Wantzen
Matrikelnummer 10 683
an der Fachhochschule Stuttgart –
Hochschule der Medien
am 19. 12. 2002

Betreuer dieser Arbeit:
Prof. Hans-Heinrich Ruta (erster Prüfer)
Prof. Wolfgang Becker (zweiter Prüfer)

Hinweis
Alle in eckigen Klammern angegebenen Zahlen beziehen sich auf Literaturverweise im Quellenverzeichnis.

Verzicht
In dieser Arbeit wird auf etwa bestehende Patente, Gebrauchsmuster oder Warenzeichen nicht gesondert hingewiesen. Wenn ein solcher Hinweis fehlt, heißt dies also nicht, dass eine Ware oder ein Warenname frei sei.

Erklärung
Hiermit versichere ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst zu haben.

Alle verwendeten Quellen sind als solche kenntlich gemacht und im Anhang ausgewiesen.

Kapitel 1 – Einführung

1.0 VORWORT ... 13
1.1 DAS MANUSKRIPT ... 14
1.2 DAS DOKUMENT ... 14
1.3 DIE DATEI ... 15
1.4 DAS FORMAT ... 15

Kapitel 2 – Kurzer Abriss über die Betriebssysteme

2.0 HINWEIS ... 19
2.1 EINLEITUNG ... 19
2.2 MACOS ... 19
2.2.1 Geschichtlicher Abriss ... 19
2.2.2 Classic MacOS 9.x ... 21
2.2.2.1 Systemarchitektur ... 21
2.2.2.2 Dateisystem ... 22
2.2.3 MacOS X ... 23
2.2.3.1 Systemarchitektur ... 24
2.2.3.1.1 *Systemkernel* ... 24
2.2.3.1.2 *Grafiksysteme* ... 24
2.2.3.1.3 *Anwendungs-/Entwicklungsumgebungen* ... 25
2.2.3.1.4 *User Interface (UI)* ... 25
2.2.3.2 Dateisystem ... 25
2.3 WINDOWS ... 27
2.3.1 Geschichtlicher Abriss ... 27
2.3.2 Windows 2000 ... 28
2.3.2.1 Systemarchitektur ... 28
2.3.2.1.1 *Hardware-Abstraction-Layer (HAL)* ... 28
2.3.2.1.2 *Kernel* ... 28
2.3.2.1.3 *Treiber* ... 29
2.3.2.1.4 *Dienste* ... 29
2.3.2.1.5 *Subsysteme* ... 29
2.3.2.2 Dateisystem ... 30
2.4 DATENAUSTAUSCH ZWISCHEN DEN WELTEN ... 31

Kapitel 3 – Kurzer Abriss über die Programme

3.1 TEXTVERARBEITUNG MS WORD ... 35
3.2 DTP-SOFTWARE ... 35
3.2.1 Quark XPress ... 36
3.2.2 Adobe InDesign ... 37

Kapitel 4 – Aufbau und Struktur von Textdateien

4.0 EINLEITUNG ... 41
4.1 ZEICHENSATZ, CHARSET, ENCODING ... 41
4.1.1 7-Bit-ASCII ... 42
4.1.2 8-Bit-ASCII und ISO 8859 ... 42

4.1.3 Unicode ... 43
4.1.3.1 ISO 10 646, UCS und Unicode ... 43
4.1.3.2 UTF-8 ... 44
4.1.4 Die Problematik der Darstellung ... 44
4.2 ZEILENENDSCHALTUNG ... 45
4.3 FORMATE ... 46
4.3.1 Standardisierte Formate ... 46
4.3.1.1 Das Rich Text Format (RTF) ... 46
4.3.1.2 Markup Languages ... 48
4.3.1.2.1 *Standard Generalized Markup Language (SGML)* ... 49
4.3.1.2.2 *Extensible Markup Language (XML)* ... 51
4.3.2 Proprietäre Formate ... 52
4.3.2.1 Adobe InDesign ... 53
4.3.2.2 Quark XPress ... 54
4.3.2.3 Microsoft Word ... 54

Kapitel 5 – Der Testlauf

5.0 EINLEITUNG ... 58
5.1 DIE TESTDATEI ... 58
5.2 DAS UNTERSUCHUNGSPROTOKOLL ... 58
5.3 DAS KONVERTIERUNGSZIEL ... 58

Kapitel 6 – Von Word in DTP

6.1 COPY & PASTE ... 63
6.1.1 Einleitung ... 63
6.1.2 Die Implementierung in den Betriebssystemen ... 63
6.1.2.1 Classic MacOS ... 63
6.1.2.2 MacOS X ... 64
6.1.2.3 Windows ... 64
6.1.3 Versuchsergebnis ... 64
6.1.3.1 Quark XPress 5 ... 64
6.1.3.2 InDesign 2 ... 65
6.2 DRAG & DROP ... 65
6.2.1 Einleitung ... 65
6.2.2 Versuchsergebnis ... 66
6.2.2.1 Quark XPress 5 ... 66
6.2.2.2 InDesign 2 ... 66
6.3 WORDFORMAT-FILTER ... 66
6.3.1 Einleitung ... 66
6.3.2 Versuchsergebnis ... 66
6.3.2.1 Quark XPress 5 ... 66
6.3.2.2 InDesign 2 ... 67

6.4 RTF-FILTER 67
6.4.1 Einleitung 67
6.4.2 Versuchsergebnis 68
6.4.2.1 Quark XPress 5 68
6.4.2.2 InDesign 2 68
6.5 NUR-TEXT-FILTER 68
6.5.1 Einleitung 68
6.5.2 Versuchsergebnis 69
6.5.2.1 Quark XPress 5 69
6.5.2.2 InDesign 2 69
6.6 XML 69
6.6.1 XML vs. XML 69
6.6.2 DTP-Programme und XML 70
6.6.2.1 Quark XPress 70
6.6.2.2 Adobe InDesign 71
6.6.3 UpCast 71
6.6.4 Versuchsergebnis 72
6.6.4.1 Quark XPress 5 72
6.6.4.2 InDesign 2 72
6.7 STEUERCODES FÜR DTP-PROGRAMME 73
6.7.1 XPress-Marken 73
6.7.2 Adobes Tagged Text 74
6.7.3 Vom Word-Format zu den Steuercodes mit VBA 75
6.7.4 tagNow-Beispielmakro 75
6.7.5 Versuchsergebnis 77
6.7.5.1 Quark XPress 5 77
6.7.5.2 InDesign 2 77

Kapitel 7 – Der optimale Weg

7.0 EINLEITUNG 81
7.1 COPY & PASTE UND DRAG & DROP 81
7.2 WORD-, RTF- ODER NUR-TEXT-IMPORT 81
7.3 XML 82
7.4 STEUERCODES DER DTP-PROGRAMME 82
7.5 ERGEBNIS 82

Kapitel 8 – Ausblick

Kapitel 9 – Anhang

9.1 ÜBERSICHT ÜBER ISO 8859 91
9.2 CODIERUNG DER SONDERZEICHEN UNTER MACOS 9.2.1 92
9.3 BACKUS-NAUR-FORM 93
9.4 XML-KONFORMER TEIL EINER INDESIGN-DATEI 94

9.5 ANFANG DER TESTDATEI ... 95
9.5.1 Als Worddatei ... 95
9.5.2 XML-codiert ... 96
9.5.3 XPress-Marken-codiert ... 102
9.5.4 Tagged-Text-codiert ... 104
9.6 QUELLTEXT DES BEISPIELMAKROS TAGNOW ... 107
9.7 VERSUCHSPROTOKOLLE ... 113
9.7.1 Quark XPress 5.0 für Windows 2000 ... 114
9.7.1.1 Copy & Paste ... 114
9.7.1.2 Drag & Drop ... 115
9.7.1.3 Word-Format ... 116
9.7.1.4 RTF ... 117
9.7.1.5 Nur-Text-Format ... 118
9.7.1.6 XML ... 119
9.7.1.7 XPress-Marken ... 120
9.7.2 InDesign 2.0 für Windows 2000 ... 121
9.7.2.1 Copy & Paste ... 121
9.7.2.2 Drag & Drop ... 122
9.7.2.3 Word-Format ... 123
9.7.2.4 RTF ... 124
9.7.2.5 Nur-Text-Format ... 125
9.7.2.6 XML ... 126
9.7.2.7 Tagged Text ... 127
9.7.3 Quark XPress 5.0 für MacOS 9.2.1 ... 128
9.7.3.1 Copy & Paste ... 128
9.7.3.2 Drag & Drop ... 129
9.7.3.3 Word-Format ... 130
9.7.3.4 RTF ... 131
9.7.3.5 Nur-Text-Format ... 132
9.7.3.6 XML ... 133
9.7.3.7 XPress-Marken ... 134
9.7.4 InDesign 2.0 für MacOS 9.2.1 ... 135
9.7.4.1 Copy & Paste ... 135
9.7.4.2 Drag & Drop ... 136
9.7.4.3 Word-Format ... 137
9.7.4.4 RTF ... 138
9.7.4.5 Nur-Text-Format ... 139
9.7.4.6 XML ... 140
9.7.4.7 Tagged Text ... 141
9.7.5 Punktevergleich ... 142
9.8 QUELLENVERZEICHNIS ... 143
9.8.1 Print-Quellen ... 143
9.8.2 Non-Print-Quellen ... 143
9.9 ABKÜRZUNGS- UND DEFINITIONSVERZEICHNIS ... 145

Kapitel 1

Einführung

1.0 VORWORT

Der im Motto angeführte Vergleich mit dem Schachspiel ist kein weit hergeholter. Zwar soll uns der Computer abnehmen und automatisieren was zuvor stoische Handarbeit erforderte, doch gleicht die Erreichung eines solchen Zieles am Rechner zuweilen eher geschicktem Taktieren – ähnlich dem des Schachspiels – als einem gelassenen Zurücklehnen in den Bürostuhl. Schach hat demgegenüber sogar noch einen Vorteil anzuführen: Man kann seinem Gegner direkt in die Augen schauen. Am Computer ist man sich zumeist nie sicher, ob das Betriebssystem, die Funktionalität oder die Benutzerschnittstelle des Programms, man selbst oder eine bunte Mischung aus all dem gegen einen arbeitet. Doch sind leider nicht nur für individuelle Spezialfälle solch komplexe Problemlösungsstrategien zu entwickeln.

Diese Diplomarbeit beleuchtet eigentlich ein Paradoxon. Microsoft Word ist die von Autoren meistgenutzte Textverarbeitungssoftware[1] und die hier untersuchten DTP-Programme sind Standards in Druckereibetrieben. Es ist also ein häufiges Ereignis, dass der Inhalt einer MS-Word-Datei in ein DTP-Programm übernommen werden muss, weil einige drucktechnisch wesentliche Aspekte[2] in der Büro-Anwendung von Microsoft nur unzureichend bis gar nicht implementiert wurden.[3] So verwundert es, dass für eine solche Eventualität noch kein immer funktionierender Standardweg existiert.

Die bedeutenden DTP-Programme sind mit einer eigenen Meta-Sprache (beispielsweise die XPress-Marken von Quark XPress) ausgestattet worden, die das Programm bei einem Text-Import interpretieren kann. Doch wie Autoren ihre Texte so erfassen können, dass sie dabei möglichst wenig behindert werden (sich also nicht zu sehr mit Systemtechnik herumschlagen müssen), der Text aber schon fix und fertig zum einfachen Übernehmen in ein vorgestaltetes Layout ist, bleibt ungewiss.

Die sicherlich einfachste Lösung wäre das Programmieren eines Texteditors, der hinter einer leistungsfähigen und einfach zu bedienenden Benutzeroberfläche eben jene Meta-Sprache oder besser noch das proprietäre, also interne, Dateiformat der DTP-Software benutzt. Solche Editoren wurden für die breite Masse nie geschrieben, hätten aber mit Sicherheit vieles stark vereinfacht. Heute findet sich dieser Gedanke in Redaktionssystemen wieder, welche von den DTP-Programmherstellern angeboten werden. Diese sind jedoch für Anwendungen außerhalb von Zeitschriften-, Zeitungs- und der massenhaften Buchproduktion schlicht zu teuer.

Auch denkbar wäre eine Kooperation mit beispielsweise Microsoft oder einem anderen Hersteller von Textverarbeitungssoftware. Doch auch dies wurde bisher nicht realisiert. Vielleicht hätte eine teilweise Offenlegung des Programmcodes (ähnlich dem Open-Source-Gedanken) zu einer Implementierung geführt, doch werden sämtliche internen Dateiformate gut gehütet und vor der Öffentlichkeit verschlossen.

Ein dritter Weg schließlich stellt die Einigung auf einen allgemein gültigen Standard (zum Beispiel einen ISO-Standard) für das Markup des textlichen Inhalts dar.[4] Und sicherlich lassen sich noch weitere Lösungsalternativen generieren, doch wurden bisher keine bis wenige Anstrengungen zur Lösung des Problems unternommen.

Diese Arbeit soll die verschiedenen derzeit möglichen Wege untersuchen, die begehbar sind, um einen Text ohne großen Nachbearbeitungsaufwand von MS Word in ein DTP-Programm zu transferieren. Die Betrachtungen werden sich ausschließlich auf textuelle Probleme beschränken, sich also nicht auf Grafik- oder Bilddaten beziehen. Die einzelnen Wege werden anhand aufgestellter Kriterien untersucht und bewertet. Daraus ergibt sich ein optimaler Weg, der abschließend begutachtet und diskutiert wird.

Den nachfolgenden Teil bilden einige Definitionen und Betrachtungen, die als inhaltlicher Unterbau der Arbeit anzusehen sind.

1 Laut einer Studie des Unternehmens Gartner Dataquest beherrscht Microsoft mit den Office-Produkten den Markt zu 95 Prozent. Das kostenlose StarOffice-Paket von Sun Microsystems wird bis zum Jahr 2004 ein auf 10 Prozent steigender Marktanteil prognostiziert. [16]

2 Als Beispiele seien hier nur wenige genannt: Umformatieren ganzer Dokumente bei Änderung des Druckertreibers, korrektes Definieren eigener Farben, Vierfarbseparation und so weiter.

3 Es gibt Druckereien, die den Source-Code von MS Word gekauft und dahingehend verändert haben, dass eine drucktechnische Verarbeitung sinnvoll möglich ist. Die Kosten hierfür sind allerdings immens.

4 Hier werden in XML große Hoffnungen gesetzt. Aktuelle Versionen verschiedener DTP-Programme können nun erstmals mit XML umgehen, was in Kapitel 6.2 untersucht wird.

1.1 DAS MANUSKRIPT

Grundstein der setzerischen Arbeit an einem Text ist das Manuskript. Im Lexikon ist folgende Definition nachzulesen: »[…] 2. Die erste schriftliche Festlegung eines Sprachwerks, gleichgültig, ob hand- oder maschinenschriftlich. M. ist auch jede sonstige Vorlage für den Setzer im Unterschied zum Vervielfältigungsstück […].« [3]

Bei seiner Arbeit steht für den Autor die geistige Substanz, nicht aber die Form, in der sie niedergeschrieben wird, im Vordergrund. So sind der physischen Gestalt von Manuskripten kaum Grenzen gesetzt – von der gedruckten Vorlage über Datenträger und das einwandfreie Schreibmaschinen-Manuskript bis hin zum vor- und rückseitig handgeschriebenen Zettel ist alles anzutreffen. Seitdem Computer Einzug in die Autorenwelt gehalten haben, hat sich die Problematik unleserlicher Handschriften in anderer Weise fortgeführt: Es existiert eine Vielzahl von Dateiformaten, die je nach Betriebssystem und verwendeter Programmversion untereinander inkompatibel sind.

Grundlegend kann auf den Arbeitsstil eines festen Autors mehr eingewirkt werden als auf den eines freiberuflichen, während man auf Gelegenheitsschreiber im Allgemeinen kaum Einfluss hat. Doch bleibt zu erwähnen, dass berufsmäßige und freiberufliche Autoren zumeist für mehrere Verlage tätig sind und sich somit immer wieder neu auf einen individuellen Workflow einstellen müssen, wozu einige nicht bereit sind.

Um jedoch kosteneffizient arbeiten zu können, müssen Verlag und Setzerei Rücksprache mit dem Autor halten – ein Punkt, der in der Praxis leider allzu häufig missachtet wird, obwohl dadurch Kosten und Zeit gespart und Fehler vermieden werden können.

1.2 DAS DOKUMENT

Der Brockhaus Multimedial [19] definiert ein Dokument im informationstechnischen Sinne als »Datei, die mit einem Anwendungsprogramm erstellt und bearbeitet wurde«, aus rechtstheoretischer Sicht als eine/ein »(mittellateinisch, zu lateinisch docere ›beweisen‹) Urkunde, amtliches Schriftstück; Beweismittel, Beleg«.

Die Definitionen lassen vermissen, was ein Dokument letztendlich zum Inhalt hat. Unter Verzweigung in die Definition des Begriffs Urkunde lässt sich dies ergänzen, denn eine Urkunde ist »jeder Gegenstand, der einen menschlichen Gedanken verkörpert (zum Beispiel Grenzstein), im engeren Sinn die schriftliche Festlegung eines Gedankens (Dokument) […]« [19]. Auch wenn hier der Brockhaus eine etwas unsaubere Begriffsbestimmung gibt, lässt sich daraus eine für diese Arbeit gültige Definition ableiten: Ein Dokument ist die mediale Festlegung von Gedanken, also eine Sammlung aufgezeichneter Informationen. Recht intuitiv lässt es sich demnach in die folgenden Grundbestandteile aufgliedern: in Inhalt, Struktur und Formatierung.[5]

Als Inhalt[6] soll der eigentliche Informationsgehalt des Dokuments, also die geistige Substanz, bezeichnet werden.

Die Struktur beschreibt die Gliederung des Inhalts in Bereiche unterschiedlicher Wertigkeit und die Beziehungen zwischen diesen Bereichen. Die Information wird in Einheiten kategorisiert, die zusammengehörig sind beziehungsweise gegeneinander abgegrenzt werden.

Formatierung[7] schließlich ist die sinnlich wahrnehmbare Erscheinung des Dokuments unabhängig des Aufzeichnungsmediums. Sie macht uns Menschen sowohl den Informationsgehalt als auch die Struktur sichtbar und plausibel. Es existieren allerdings auch Informationen, die mittels Format nicht, wohl aber strukturell weiter unterschieden werden können.

Die scharfe Abgrenzung zum Manuskript ist definitionsgemäß nur schwer zu fassen, so dass ein Umweg über die

5 Man könnte noch die Metainformationen als einen Grundbestandteil definieren. Da sie aber keinen sichtbaren Einfluss auf die Ausgabe eines Textes haben, werden sie bei dieser Betrachtung außen vor gelassen.

6 Abweichend von der Literatur und Sekundärliteratur (vergleiche [7]) verwendet diese Darstellung den Begriff »Inhalt« statt des Begriffs »Daten«, um stärker herauszuheben, dass es sich um die »geistige Substanz« des Dokuments handelt und damit etwas weiter vom Computerumfeld distanziert ist (von »Daten« spricht man im Zusammenhang mit Computern umgangssprachlich auch von Dateien, die auf einem Massenspeicher abgelegt sind).

7 Auch hier wird explizit auf den Begriff »Format« verzichtet, weil er in der Computersprache mit »Dateiformat« gleichgesetzt wird.

Wortsemantik nötig wird: Liegt der konnotative Schwerpunkt beim Manuskript eindeutig auf der ersten schriftlichen Fixierung, so ist beim Dokument der Fokus auf den Inhalt, die geistige Substanz gerichtet.

1.3 DIE DATEI

Um den Begriff der Datei eingrenzen zu können, muss zunächst betrachtet werden, aus was Dateien bestehen: Daten. Diese sind nach dem dtv-Lexikon wie folgt definiert: »Informatik: Zeichenfolgen (digitale D.) oder kontinuierliche Funktionen (analoge D.), die Objekte für den Arbeitsprozess einer D.-Verarbeitungsanlage sind: Sie werden im Computer ›maschinell‹ verarbeitet, dann durch Peripheriegeräte (Datensichtgeräte, Drucker) als ›lesbare‹ Informationen (Texte, Graphiken) ausgegeben. Vor ihrer Verarbeitung mit dem Computer müssen digitale D. auf einem maschinenlesbaren Datenträger in codierter Form bereitgestellt werden.« [2]

Demnach codieren Daten also Informationen.

Zum Begriff Datei vermerkt das gleiche Lexikon: »EDV: geordnete Mengen maschinenlesbarer Daten auf Datenträgern (zum Beispiel Magnetbänder und -platten) oder in internen Speichern, die über einen D.-Namen angesprochen werden können und i. A. vom Betriebssystem der Datenverarbeitungsanlage verwaltet werden; sie können sowohl Programme als auch nur die zu verarbeitenden Daten enthalten.« [2]

Zusammenfassend und vereinfachend lässt sich definieren: Dateien sind Sammlungen codierter Informationen (Daten), die von Datenverarbeitungssystemen verwaltet werden. Eine weitere Abgrenzung der verschiedenen Dateitypen erfolgt in Kapitel 1.4 und 4.

1.4 DAS FORMAT

Laut wissen.de [37] ist der Begriff Format aus informationstechnischer Sicht eine »Beschreibung des Aufbaus bzw. der Anordnung von Daten, Datenträgern und Programmen. Das Datenformat eines Rechners legt die Art fest, wie die im Hauptspeicher als Bitfolgen abgelegten Daten bei der Bearbeitung durch ein Programm interpretiert werden sollen […]«. Demnach ist ein Dateiformat die innere logische Struktur, der innere Aufbau einer Datei. Zu jedem Dateiformat existiert somit ein Regelwerk, das dessen Aufbau festschreibt.

Man unterscheidet zwischen standardisierten und nicht standardisierten, so genannten proprietären Dateiformaten. Gemeint ist damit allerdings nicht, dass die proprietären Formate keinen definierten inneren Aufbau hätten, sondern das Maß an Öffentlichkeit. Ein standardisiertes Format ist ein öffentlich festgelegtes Format, das jedem zugänglich ist. Im Gegensatz dazu steht das proprietäre Format: Der DUDEN definiert einen Proprietär als »Eigentümer« [4]. Demnach hält ein Hersteller an einem proprietären Format Eigentumsrechte. Diese Formate sind meist nicht öffentlich zugänglich. Eine detailliertere Unterscheidung nimmt Kapitel 4.3 vor.

Kapitel 2

Kurzer Abriss über die Betriebssysteme

2.0 HINWEIS

Ein Blick auf Aufbau, Architektur und Arbeitsweise der Betriebssysteme ist unerlässlich um die in Kapitel 6 untersuchten Mechanismen verstehen und bewerten zu können. Doch beschränken sich die Ausführungen stark auf das Thema dieser Arbeit. Für tiefer gehende Informationen sei auf die Quellen im Anhang verwiesen.

2.1 EINLEITUNG

Betriebssysteme wie Linux und UNIX finden sich in der grafischen Industrie seit dem Verschwinden der Berthold-Workstations nur mehr als File- beziehungsweise Produktions-Server. Die umfangreichen und detaillierten Konfigurationsmöglichkeiten dieser Systeme sind Segen und Laster zugleich. Ein System, das ausführlich für den Verwendungszweck konfiguriert und kompiliert wurde, ist optimal darauf ausgerichtet und läuft extrem stabil. Doch ist der Prozess dorthin komplex und bedarf tiefer Einblicke in die Struktur von Hard- und Software, was von einem Anwender nicht erwartet werden kann. Ebenso steht das Phänomen der Komplexität im krassen Gegensatz zur Simplizität, die von DTP-Programmen und grafischen Benutzeroberflächen propagiert und auch gefordert wird (vergleiche Kapitel 3.2). Es gibt viele erfolgreiche Versuche, auf ein Linux-System eine grafische Oberfläche zu programmieren, doch findet das System erstens wenig Gegenliebe unter den Programmierern, die maßgeblich Software für die grafische Industrie schreiben, und zweitens unter den Benutzern, die damit arbeiten sollen, obwohl viele Punkte für Linux sprächen.

Die Kompatibilität der Systeme zwischen Autoren und Setzerei ist wichtig, um die ohnehin komplexen und komplizierten Konvertierungsvorgänge wenigstens etwas zu vereinfachen. Die meisten Autoren arbeiten unter Microsoft Windows. In vielen Setzereien sind Macintosh-Systeme installiert. Die Ursache für diesen Widerspruch – immerhin erschweren sich die Setzereien die Text-Umsetzung somit selbst – liegt in der Herkunft der DTP-Programme begründet (vergleiche Kapitel 3.2).

So konzentriert sich die vorliegende Arbeit auf genau dieses aus der Praxis vorgegebene Spannungsfeld und untersucht einzig MacOS und Windows.

2.2 MACOS

2.2.1
Geschichtlicher Abriss

Das MacOS ist eng verwoben mit der Apple Hardware. Während Windows sämtliche Intel-basierten Prozessoren unterstützt und Microsoft damit von Beginn an ein reines Software-Unternehmen war, setzte Apple seit jeher neben der System-Software auch auf die eigene Hardware-Produktion. Was auf der einen Seite als Vorteil gesehen werden kann, sind doch Betriebssystem und Hardware als ganzes sehr gut aufeinander abgestimmt, kann auf der anderen zum Nachteil werden. Die Verbreitung von MacOS wäre um ein vielfaches höher, wenn das System auch auf Intel-Hardware – und damit dem Großteil aller installierten PCs – lauffähig wäre.

Am 1. April 1976 wurde die Apple Computer Company von Steve Wozniak, Steve Jobs und Ron Wayne gegründet. Die Geschichte des MacOS begann mit dem BASIC-gesteuerten Computer Apple I. Die Apple Computer Inc. wurde am 3. Januar 1977 offiziell eingetragen. Die ersten Schritte zu einem Betriebssystem mit grafischer Benutzeroberfläche (GUI – Graphical User Interface) basierten auf den Arbeiten zum 1973 fertig gestellten Alto des Xerox PARC[8] Institut. Das Projekt trug den Namen Lisa und wurde 1983 mit dem LisaDesk als System und dem Programmpaket MacWorks verkauft. Zeitgleich wurde der Apple IIe vorgestellt, einer der bekanntesten und erfolgreichsten Computer der Apple-Geschichte; er wurde nahezu zehneinhalb Jahre produziert.

1984 wurde System 1.0 veröffentlicht: 216 KB groß, inklusive des 42 KB großen Finders. Der Macintosh gewann den Industrial Design Excellence Award. Ein aufwändiger Werbespot, der sich an George Orwells 1984 anlehnte, wurde ein großer Erfolg.

8 Palo Alto Research Center: Forschungszentrum der Kopiererfirma Xerox – oder auch doppeldeutig: Park.

Am 22. November 1985 unterschrieb Apple CEO John Sculley einen Vertrag mit Bill Gates, der es Microsoft erlaubte, Apple-Technologie in Windows zu benutzen, wenn das Unternehmen weiterhin Software für das MacOS produziere. Steve Jobs verließ Apple und gründete die Firma NeXT Inc. In diesem Jahr erschien mit dem System 2.2 die erste Version des LaserWriter-Treibers.

Fast zeitgleich mit der Veröffentlichung des Apple IIx kam System 6.0, das mit dem Multifinder einen bedeutenden Entwicklungsschritt machte. Mit System 7.0 wurde 1991 kooperatives Multitasking und die 32-Bit-Architektur eingeführt. Es unterstützte die damals aktuellen Apple-II-Modelle (IIcx, IIfx und IIsi), den 1990 eingeführten Classic und den LC. 1991 startete Apple die Quadra- und Powerbook-Reihe, die je nach Konfiguration die neuesten Motorola-Prozessoren, den 68030 oder den 68040, enthielten. Mit System 7.1 erhielt das MacOS einen eigenen Schriften-Ordner und die QuickTime-Erweiterung. 1992 wurde die Performa-Reihe gestartet. AppleScript fand Einzug in System 7.1.1.

Ende 1994 wurden die ersten PowerMacs und das neue System 7.5 verkauft. In den Versionen 7.5.1, 7.5.2 und 7.5.3 starteten heute so wichtige Komponenten wie Open Transport, OpenDoc und die MacOS Runtime for Java (MRJ), die Grundbausteine für das TCP/IP-Protokoll und den Internetzugang sind.

1994 ist auch das Jahr der Macintosh-Clones. Apple lizenzierte das MacOS und die ROM-Routinen an andere Hersteller, die ab 1995 eigene Computersysteme verkauften. Die Clone-Ära endete 1998.

Die nächsten großen Änderungen kamen 1997 mit System 7.6 und anschließend System 8.0 einher. Das grafische Interface hatte sich schon mehrfach gewandelt und zeigte nun dreidimensionale Icons. Viele vorher fest einprogrammierte Features konnte der Benutzer frei konfigurieren. Steve Jobs stieg bei Apple als Berater wieder ein und löste innerhalb kürzester Zeit Gil Amelio als CEO ab.

Apple begann das Projekt Rhapsody, ein völlig neu konzipiertes und programmiertes Betriebssystem, das allerdings am Widerstand der Softwarehersteller scheiterte, da sämtliche Software hätte neu programmiert werden müssen. Mit System 8.1 überarbeitete Apple sein File-System – HFS+ löste mit verbesserter Nutzung der Speichermedien langsam das alte HFS (Hierarchical File System) ab. System 8.1i integrierte USB- und Firewire-Unterstützung, während mit 8.5 zum ersten Mal das Suchprogramm Sherlock auf den Markt kam.

Mit dem G3 und dem iMac gelang Apple 1998 nach einigen unsicheren Jahren wieder der große Durchbruch. Produktdesign gehörte ab sofort als Verkaufsargument wieder zur Unternehmensstrategie.

1999 wurde das neue Betriebssystem MacOS X Server vorgestellt. Es lief auf den Workgroup Servern und basierte auf dem UNIX-System OpenBSD. MacOS X Server unterstützte Adobe Display PostScript und wurde mit dem Standard-Web-Server Apache ausgestattet. Parallel entwickelte sich das MacOS 9.0. Es führte eine Mehr-Benutzer-Verwaltung ein, enthielt die neue Version 2.0 des Sherlock und die Web-Sharing-Erweiterung, die einfache Web-Server-Dienste ermöglichte.

Mitte 2000 erschien die Beta-Release von MacOS X. Der Kernel mit dem Codenamen »Darwin« basierte auf einer Fusion des FreeBSD-Mach-3-Kernels und dem NeXT-System. Es enthielt die Carbon- und Cocoa-Librarys und MacOS 9.04 als erste Classic-Umgebung. Das User-Interface erfuhr eine grundlegende Überarbeitung mit dem Aqua-Interface, Sherlock ging in die dritte Generation. Viele Software-Hersteller kündigten Carbon-Versionen ihrer Programme an. In kurzer Folge wurden in 2001 System 9.1, System X 10.0, 9.2 und die ersten Updates 10.0.1 bis 10.0.4 ausgeliefert. Sie unterstützten nun DVD, AirPort und viele weitere Peripherie-Geräte. Im Jahr 2002 ging MacOS X in die Version 10.2. Steve Jobs trug System 9.2.2 auf der Entwicklerkonferenz symbolisch zu Grabe – 9.2.2 ist damit die letzte Version der alten Systemreihe.

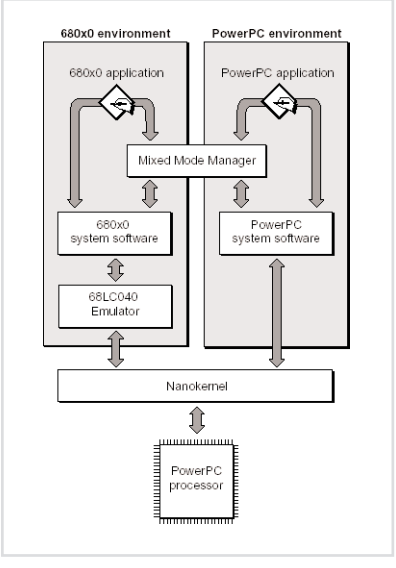
Seit Beginn ist das MacOS strikt an die Macintosh-eigene Hardware gebunden. Auch im Zeitalter des Mac-OS X hat sich daran noch nichts geändert: Der FreeBSD-Kernel kann zwar auf Intel-Prozessoren angepasst compiliert werden, doch scheut Apple noch immer vor einer endgültigen Portierung zurück.

2.2.2
Classic MacOS 9.x
2.2.2.1
Systemarchitektur

Sämtliche Dokumentation zu MacOS 9.x bezieht sich ausschließlich auf die Programmierung von Anwendungen. Keine befasst sich mit dem technischen Aufbau des Systems, weshalb hier eine Gliederung der ansonsten ausführlich dokumentierten APIs[9] selbst vorgenommen werden muss.

Die erste System-Ebene ist der so genannte Nanokernel. Er stellt die Schnittstelle der Systemsoftware zum Mikroprozessor dar, wandelt also Befehle in Prozessorcode um.

Um Abwärtskompatibilität zu garantieren, laufen quasi zwei Betriebssysteme parallel im MacOS 9.x. Das 680x0-Environment ist die Laufumgebung für Programme, die für 680x0-Prozessoren programmiert wurden. Lauffähig sind alle Programme ab der Systemsoftware 7.1. Programme, die für MacOS 8.1 und spätere Versionen programmiert und damit auf den PPC-Chip optimiert wurden, laufen im PowerPC-Environment. Da einige Programme Librarys aus beiden Environments nutzen, existiert ein Mixed-Mode-Manager, der das Wechseln zwischen beiden Systemen überwacht und leitet.



Der Systemaufbau gleicht keinem anderen bekannten System. Die wichtigste Datei auf der Festplatte ist das »System« im Systemordner. Sie enthält die mit Abstand wichtigsten Funktionalitäten und stellt den Kern des Betriebssystems mit seinen verschiedenen Komponenten und Diensten dar. In dieser Arbeit wird das PowerPC-Environment in seinen Grundzügen dargestellt, da die meisten Applikationen darauf optimiert sind.

Unter dem Begriff »Manager« fasst Apple eine Gruppe von APIs zusammen, die inhaltlich zusammen gehören. Die Datei »System« beinhaltet eine Vielzahl von Managern, die im Nachfolgenden – zu Gruppen zusammengefasst – kurz erläutert werden sollen.

Systemdienste:

- *Process Manager:* verteilt die Prozessor-Zeit auf die verschiedenen Prozesse.
- *Thread Manager:* verwaltet verschiedene Threads[10] eines Programms.
- *Event Manager:* verwaltet die auftretenden Events.
- *Memory Manager:* verwaltet Hauptspeicher-Anfragen.
- *Virtual Memory Manager:* verwaltet den virtuellen Speicher.
- *Mixed Mode Manager:* verwaltet das Springen zwischen 68k- und PPC-Instruktionen.
- *Code Fragment Manager:* Fügt Code-Fragmente aus Librarys und der Applikation zu einem lauffähigen Code im Hauptspeicher zusammen. Das System entspricht in seiner Funktionalität dem der DLLs (Dynamic Link Librarys) unter Windows.
- *Multitasking/Multiprocessing:* Multiprocessing ermöglicht Applikationen die Nutzung mehrerer Prozessoren in einem System, während Multitasking die zur Verfügung stehende Prozessorzeit aufteilt, so dass mehrere Applikationen quasi gleichzeitig auf einem Prozessor laufen und arbeiten können.
- *Program to Program Communications Toolbox:* verwaltet die Kommunikation zwischen Prozessen.
- *Notification Manager:* ermöglicht das Empfangen von Apple-Events für Programme, die im Hintergrund laufen.

9 APIs sind so genannte Schnittstellendefinitionen (Application Programming Interfaces), die vom Betriebssystem- oder Programmhersteller zur Verfügung gestellt werden. Mit ihnen kann ein Programmierer System- oder Programmfunktionen direkt ansprechen und für seine Zwecke nutzen.
10 Threads sind kleine Subprogramme, die im Speicher verteilt laufen und von ihrem Hauptprogramm genutzt werden.

Ein- und Ausgabedienste:

- *Device Manager:* Interface zwischen Applikationen und diversen Geräten.
- *Sound Manager:* verwaltet die Sound-Ausgabe und -Eingabe.
- *Communications Toolbox:* ist für alle Arten der Kommunikation zwischen Komponenten (Ansprechen des Serial Ports, Druckerschnittstelle und Ähnliches) verantwortlich.

Grafiksystem:

- *QuickDraw:* Grafik-System für die Darstellung von 2D-Grafiken auf dem Bildschirm und andere Rasterungen.
- *Window Manager:* Dieser Manager ist für die Darstellung der Fenster verantwortlich.

Dateisystem-Dienste:

- *File System Manager:* Kommunikation mit fremden Dateisystemen.
- Dienste für das Laden und Speichern von Daten mit Speichermedien.
- *Folder Manager:* für die Verwaltung der speziellen Ordner (wie Systemordner, Desktop Folder und so weiter).

GUI-Dienste:

- *Dialog Manager:* Benutzer-Interaktionen mit Dialogen.
- *Menu Manager:* Menu-Bars und Menüs.
- *Control Manager:* verwaltet alle Kontrollelemente wie Buttons, Scroll Bars und so weiter.
- *Standard File:* stellt ein einheitliches UI für Öffnen und Speichern zur Verfügung.
- *Navigation Services:* Implementierung der neuen Öffnen- und Speichern-Dialoge.
- *Scrap Manager:* verwaltet Zugriffe auf die Zwischenablage (Clipboard).
- *Drag Manager:* Drag-and-Drop-Operationen.

Text-Dienste:

- *TextEdit:* der alte Editorbefehlssatz. Das Programm SimpleText vermittelt einen guten Eindruck, was er zu leisten vermag.
- *Multilingual Text Editor:* stellt neue Editor-Routinen für Unicode-Text-Dokumente zur Verfügung.
- *Unicode Text Utilities:* Stellt Routinen für Unicode-Texte zur Verfügung.
- *ATSUI (Apple Type Services for Unicode Imaging):* Routinen zum Rendern von Unicode-Text.
- *Font Manager:* bereitet die Font-Glyphen für die Darstellung auf.

Zwei weitere unbedingt notwendige Dateien sind das »MacOS ROM File« und die »System Ressources«. Routinen, die früher in einem echten ROM-Baustein auf der Hauptplatine fest eingebrannt waren, sind seit der G3-Bauweise in eben diesen Dateien hinterlegt (der Chip auf der Platine existiert nicht mehr). Sie enthalten wichtige grundlegende Funktionalitäten wie die Standard C Library (StdCLib), eine Bibliothek, die für alle C-Programme zur Verfügung stehen muss. Des Weiteren viele BIOS-Informationen (Basic-Input/Output-System) wie der grundlegende Aufbau von Bildschirmen und Massenspeichern, MacOS-Medien-Support (wie beispielsweise MacOS USB Support) und vieles mehr.

Die letzte wichtige Datei ist der »Finder«. Er stellt ein Interface für den User und Systemereignisse zur Verfügung, mit dem Zugriffe auf das Dateisystem verwaltet werden können.

Die Konfiguration einiger Systemkomponenten wird über die Kontrollfelder (Control Panels) ermöglicht. Um die Funktionalität des Betriebssystems noch um viele wichtige Standards (ISO 9600, UDF, OpenGL, QuickTime und weitere) und nützliche Funktionalitäten, die nicht direkt ins System eingebaut sind, sondern damit zusammen arbeiten[11], zu erweitern, gibt es die Möglichkeit, so genannte Systemerweiterungen (System Extensions) zu integrieren.

2.2.2.2 Dateisystem

Das Format, mit dem der Macintosh die Dateien abspeichert heißt Mac OS Standard. Der technische Begriff hierfür lautet HFS (Hierarchical File System). Um größere Festplatten zu unterstützen hat Apple mit System 8.1 ein

11 Wie AppleScript, ColorSync, GameSprockets, LaserWriter, MacOS Runtime for Java, TCP/IP, OpenTransport, Text Encoding Converter (TEC) und die wichtigen Carbon Libraries, die es MacOS 9.x ermöglichen, OS-X-fähige Carbon-Programme laufen zu lassen.

neues Dateiformat eingeführt: HFS+. Beide Varianten verwenden den Doppelpunkt (:) als Trennzeichen (Delimiter), um in Pfadnamen die einzelnen Elemente voneinander abzugrenzen.

Jedes File erhält eine eindeutige Nummer zugeordnet, die FileID. Somit können die Namen von Ordnern, Festplatten, Dateien geändert werden, ohne dass jeweils eine – unter Umständen riesige – Liste von Pfadnamen erneuert werden muss. Bewegen und Kopieren von Dateien geht einfacher und schneller.

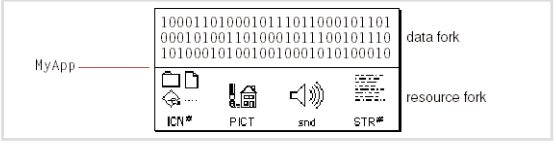
Dateinamen dürfen nur bis zu 31 Zeichen lang sein und nutzen unter HFS+ das UTF-16-Encoding, während HFS das Mac-Standard-Encoding verwendet. Auch wenn das File-System den Dateinamen in der eingegebenen Groß- und Kleinschreibung abspeichert (case-preserving), beachtet es sie nicht (und ist damit nicht case-sensitive). Demnach ist es nicht erlaubt, in ein und demselben Ordner eine Datei »Diplomarbeit.txt« und eine Datei »diplomarbeit.txt« abzulegen oder zu erzeugen.

Im Gegensatz zu HFS kann HFS+ Metadaten mit den Dateien abspeichern. Diese werden beispielsweise von UNIX-Systemen verwendet um bestimmte Eigenschaften wie Sichtbarkeit oder Schreib- und Leserechte zu speichern.

Um eine Datei einer bestimmten Anwendung zuzuweisen existieren beim MacOS zwei vierstellige eindeutige Codes: der File- und der Creator-Type. Diese sind für den Anwender unsichtbar und mit Programmen wie ResEdit veränderbar. Der File-Type beschreibt die Art einer Datei (»JPEG« für JPEG-Dateien, »TEXT« für reine Textdateien und so weiter), der Creator-Type das Programm, welches die Datei erstellt hat. Bei einem Doppelklick auf die Datei wird dann dieses Programm geöffnet, um die Datei anzuzeigen. Ist das entsprechende Programm nicht auf dem Rechner installiert, kann der Anwender über ein Dialogfeld entscheiden, mit welchem Programm er die Datei öffnen will. Der Creator-Type definiert auch, mit welchem Icon die Datei im Finder angezeigt wird. Werden Dateien von anderen Betriebssystemen mit Dateiendungen übernommen, kann das MacOS über das Kontrollfeld FileExchange einen File-Type und -Creator zuordnen.

Die klassischen Mac-Betriebssysteme speichern – im Gegensatz zu MacOS X, Windows und UNIX-Systemen – die Daten in so genannten komplexen Files oder Multi-Fork-Files.

Jedes File besteht aus zwei Forks (Zweigen): dem Resource- und dem Data-Fork (zu Deutsch Ressourcen- und Daten-Zweig). Der Data-Fork enthält die Hauptdaten einer Datei, also zum Beispiel den ausführbaren Code eines Programms oder den expliziten Inhalt. Der Resource-Fork enthält vielschichtige Hilfsdaten, die ein Programm zum Funktionieren benötigt oder eine Datei näher beschreiben. Das sind zum Beispiel Metainformationen wie Versionsnummer und Speicherbedarf, aber auch Angaben, wie Menüs aussehen sollen, Text und Layout von Warn- und Fehlermelde-Fenstern, Programmicons, Bilder, Sounds. Der versierte Benutzer hat mit dem Freeware Programm ResEdit von Apple die Möglichkeit, die Ressourcen eines Programms zu verändern.



Über Systemerweiterungen kann der System 9.x noch Medien mit UDF und ISO 9660 verwalten.

2.2.3 MacOS X

Wie Microsoft mit Windows NT (vergleiche Kapitel 2.3) entschied sich Apple, ein grundlegend neu programmiertes, modernes Betriebssystem zu entwickeln. Das Optimieren eines alten Codes unter gleichzeitiger Beibehaltung weitreichender Kompatibilität verlangte viele Kompromisse, die der Benutzer zumeist mit Abstürzen und sonstigen Fehlermeldungen bemerkte. Die starke Verwobenheit der Betriebssystemkomponenten ermöglichte einen kompletten Systemabsturz, auch wenn nur ein kleines unbedeutendes Teilmodul betroffen war.

Apples erster Versuch, das MacOS neu zu programmieren wurde 1997 durch die Applikationshersteller niedergeschmettert: Rhapsody verschwand wieder. Die

Portierung der bestehenden Software war nicht zu bewerkstelligen, zu weit war das System vom damaligen MacOS 8.x entfernt. Alle Applikationen hätten völlig neu programmiert werden müssen – ein Aufwand, den niemand bezahlen wollte.

Apple entschied sich für einen anderen, »sanfteren« Weg. Jobs kaufte seine alte Firma NeXT und konzipierte auf der UNIX-Grundlage FreeBSD das neue System MacOS X.

Damit vereinte Apple die stabile und moderne Architektur des UNIX-Kernels mit den wichtigsten Grafiktechnologien und einem komplett neu entwickelten User-Interface.

Ab Januar 2003 werden alle neuen Macs nur noch mit MacOS X ausgeliefert. Die Tage des Classic MacOS sind gezählt.

2.2.3.1 Systemarchitektur

2.2.3.1.1 *Systemkernel*

Die Grundlage des MacOS X bildet das Open-Source-System Darwin. Es integriert eine Reihe von Technologien, unter anderem den Mac-3.0-Kernel, Dienste für Betriebssysteme, die auf BSD-UNIX (Berkeley Software Distribution) basieren, leistungsstarke Netzwerkfunktionalität sowie Unterstützung für mehrfach integrierte Dateisysteme. Durch seine modulare Bauweise ermöglicht es Entwicklern das dynamische Laden von Gerätetreibern, Netzwerkerweiterungen und neuen Dateisystemen.

Das erweiterte Speichermanagementsystem bringt verbesserten Speicherschutz und damit hohe Stabilität. Jede Anwendung und jeder Prozess erhält einen eigenen, vor dem Zugriff anderer Applikationen geschützten Adressbereich. Der Speichermanager kann mehrere separate Anwendungsumgebungen verwalten und ermöglicht somit mehreren Benutzern gleichzeitig auf dem System zu arbeiten. Damit ist MacOS X ein echtes Multi-User-System.

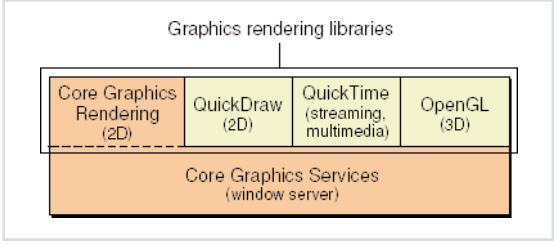
In diesem Zusammenhang sei das präemptive und kooperative Multitasking und das symmetrische Multiprocessing (SMP) erwähnt.

Gerätetreiber werden mithilfe des objektorientierten I/O-Subsystems erstellt. Die so entwickelten Treiber sorgen für Plug-and-Play, dynamisches Gerätemanagement (Hot-Plugging) und Powermanagement. Das I/O-Subsystem ist für sämtliche Ein- und Ausgabe-Prozesse zuständig.

Netzwerkprotokolle basieren auf der Network Kernel Extension (NKE).

2.2.3.1.2 *Grafiksysteme*

MacOS X kombiniert Quartz, OpenGL und Quicktime und implementiert damit wichtige Grafiktechnologien direkt im System.



Quartz ist zuständig für das 2D-Grafik-Rendering, Anti-Aliasing und das Compositing von PostScript-Grafiken. Ferner stellt es Routinen für Rendering und Drucken, die als internes Darstellungsmodell das PDF nutzen. Damit wird das Einbetten und Bearbeiten von PDF-Daten in jeder MacOS-X-Applikation möglich. Es ergeben sich einige Vorteile, wie automatische PDF-Generierung, Speichern der Dateien als PDF, eine automatische Vorschau der Grafiken am Bildschirm und die Konvertierung von PDF-Daten zu Druckrasterdaten oder PostScript – und all das direkt vom System bereit gestellt.

Quartz beherbergt auch grundlegende Dienste wie den Window-Server (in den Core Graphics Services), Cursorroutinen und das Event-Handling. Grafiken und Texte können direkt mit Anti-Aliasing versehen werden.

QuickDraw steht weiterhin für 2D-Renderings zur Verfügung. Es stellt sozusagen das Interface für Carbon-Programme dar, um auf Quartz-Routinen zugreifen zu können. Für das Rendering von 3D-Objekten ist OpenGL zuständig. QuickTime wurde für alle multimedialen Aktionen implementiert. Es unterstützt – neben seinen eigenen – nativ unzählige Bild- (PICT, BMP, GIF, JPEG, TIFF und PNG), Video- (AVI, AVR, DV, M-JPEG und MPEG-1) und diverse Streamingformate (HTTP, RTP und RTSP).

2.2.3.1.3

Anwendungs-/Entwicklungsumgebungen

Darwin kann mehrere Anwendungsumgebungen gleichzeitig bearbeiten, was die Kompatibilität zu früheren MacOS-Versionen erst ermöglicht. Die MacOS-X-Umgebungen Classic und Carbon wurden speziell für diese Art der Kompatibilität entwickelt.

Die Classic-Umgebung ist eine komplette Version von MacOS 9.2.2 und läuft in einem geschützten Speicherbereich unter MacOS X. Damit wird es möglich, Applikationen, die auf MacOS 8.1 oder höheren Systemen laufen würden, direkt im MacOS-X-Betrieb einzusetzen.

Um Entwicklern den Umstieg auf MacOS X zu erleichtern, wurde die Carbon-Umgebung geschaffen. Obwohl mit den Carbon-APIs die Vorteile der MacOS-X-Funktionen wie Multiprocessing, Aqua-UI (User Interface), erweitertes Sicherheitsmodell und Speicherschutz angesprochen werden können, wurde es speziell aus Gründen der Kompatibilität mit älteren OS-Versionen entwickelt. Die Portierung einer Classic- in eine Carbon-Anwendung ist so wenig aufwendig, da die Carbon-APIs auf früheren Versionen von MacOS-APIs basieren. Carbon-Programme gelten schon als MacOS-X-Applikationen, obwohl sie in Wahrheit ein Zwischenschritt zur echten, nativen OS-X-Applikation darstellen: den Applikationen, die auf den Cocoa-APIs basieren. Sie haben vollen direkten Zugriff auf alle relevanten Systemteile.

Des Weiteren hat Apple die komplette Java 2 Standard Edition (J2SE) implementiert. Der Java-Bereich greift durch spezielle Routinen auf wichtige Systembereiche wie Quartz zu. Damit wird die Geschwindigkeit von Applets und Java-Anwendungen erhöht, unter Beibehaltung der wichtigen Portabilität von Java.

Zu guter Letzt ist es möglich, UNIX-basierte Anwendungen auf die Plattform zu übertragen und lauffähig zu machen. Hierzu ist ein Kompilierungsvorgang unter MacOS X notwendig.

2.2.3.1.4

User Interface (UI)

Als Benutzeroberfläche hat Apple mit Aqua eine komplette Neuentwicklung realisiert. Während Apple das Look-and-Feel als das Ultimo der UI-Entwicklung hinstellt, stehen viele MacOS-Anwender dem sehr kritisch gegenüber. Viele Funktionen, wie das Schrumpfen und Wachsen der Fenster aus dem Dock, werden als Ressourcen-fressende Spielereien empfunden. Die Tauglichkeit der Oberfläche wird sich in den nächsten Jahren erst noch beweisen müssen.

Das parallele Ausführen mehrerer Anwendungsumgebungen und Dateisysteme sind in der Theorie schön zu lesen. Doch das UI entscheidet letztlich über die Benutzbarkeit. Hier sei nur vermerkt, dass es manchmal nicht einfach ist, den Überblick über die unterschiedlichen Teile und Bereiche zu behalten, wie im nächsten Kapitel etwas ausführlicher berichtet wird.

2.2.3.2 Dateisystem

Aufgrund der BSD-Erweiterungen und einer verbesserten Virtual-File-System-(VFS-)Struktur verwendet die Dateisystemkomponente von Darwin eine mehrschichtige Architektur, in der Dateisysteme stapelbar sind (also mehrere Dateisysteme gleichzeitig geladen und dynamisch im laufenden Betrieb wieder entladen werden können). Sie verfügt weiterhin über Berechtigungen an entfernbaren Medien einschließlich USB- und FireWire-Geräten, URL-basiertes Volume Mounting und auf UTF-8 basierende Dateinamen, die bis zu 255 Zeichen

lang sein können. Die Zuordnung von Datei zur Anwendung geschieht UNIX-konform über die Dateiendung. Der Separator für Dateipfade ist der Slash (/). Das System übersetzt automatisch zwischen dem Doppelpunkt älterer MacOS-Programme und -Dateien und dem Slash.

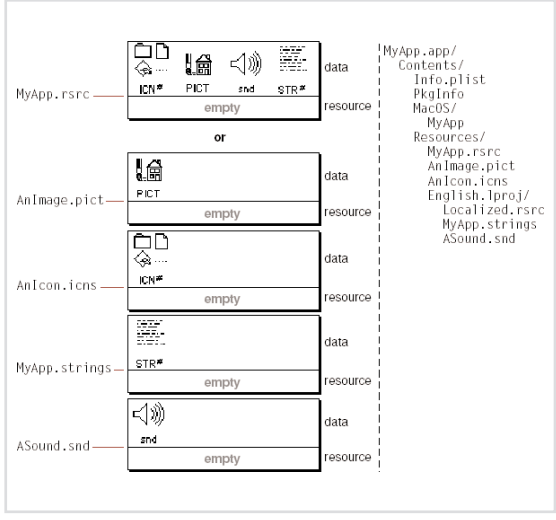
Im UFS werden keine Aliasse mehr unterstützt. Sie werden durch die UNIX-konformen Symbolic-links abgelöst, die im Handling ähnlich sind.

MacOS X unterstützt mehrere Dateisysteme: HFS, HFS+, UFS, UDF, ISO 9660, NFS und AFP:

- *HFS:* Hierarchical File System (auch: Mac OS Standard Format). Das Standard-Format des MacOS vor System 8.1.
- *HFS+:* Hierarchical File System Plus (auch: Mac OS Extended Format). Das Standard-Format von MacOS 8.1 bis 9.x.
- *UFS:* Universal File System. Basiert auf dem 4.4BSD FSF (Fast File System) und gilt als Standard für viele UNIX-Systeme.
- *UDF:* Universal Disk Format für DVD-Medien.
- *ISO 9660:* Das Standard-Format für CD-ROMs.
- *NFS:* Network File System, das vorherrschende File-Sharing-Protokoll der UNIX-Welt.
- *AFP:* Apple File Protocol, das File-Sharing-Protokoll unter MacOS 9.x.

Für den Benutzer verhält sich das System allerdings so, als gäbe es nur ein einziges File-System. In einer heterogenen Arbeitsumgebung, in der noch alte Classic-Anwendungen, aber auch neue Carbon- oder Cocoa-Programme genutzt werden, kann dies teilweise zu Verwirrungen führen. Ein HFS+-Volume unterliegt natürlich noch seinen speziellen Limitationen (vergleiche 2.2.2.2), wie auch ein Classic-Programm, so dass eine klarere Trennung der unterschiedlichen Formate – zumindest für den Übergang – doch wünschenswert wäre.

Um Dateien von alten MacOS-Versionen verarbeiten zu können, beherrscht MacOS X das HFS- und HFS+-Format. Komplexe Dateien mit Resource- und Creator-Type kann das System zwar weiterhin verwalten, doch wurde das Modell für MacOS-X-Dateien aufgegeben. Die Ressourcen werden nun als eigenständige Dateien mit der Endung .rscr (oder .dfont oder .icns und so weiter) gespeichert. Um die große Anzahl von Dateien, die mit dem Verzicht auf den Ressourcen-Zweig anfallen, einfach zu verwalten, wurden die Bundles (englisch für Bündel, Paket) erfunden. Sie sind einerseits Ordner und andererseits Alias (Verknüpfung) und lassen sich wie einzelne Dateien hin und her kopieren. Sämtliche MacOS-X-Programme sind in solche Bundles gepackt. Doppelklickt man im Finder auf ein Programmicon (das ja eigentlich ein Bundle ist), wird dieser Doppelklick an das Programm weitergegeben. Das Bundle funktioniert also wie ein Alias. Es enthält aber gleichzeitig sämtliche Dateien, die das Programm benötigt: den ausführbaren Code, die Ressourcen und die Lokalisierungsdateien (verwendete Sprache, verwendetes Zahlensystem und so weiter):



Ein einziges Programmbundle kann somit verschiedene Programmversionen für verschiedene Plattformen und Volumeformate enthalten (so kann es auf MacOS 8, 9 und X laufen). Ein Client kann ein Programm problemlos auf einem Server laufen lassen und einfache Programme, die nur aus einem einzigen Paket bestehen, können per Drag & Drop installiert und ausgetauscht werden – um nur einige Vorteile zu nennen.

2.3 WINDOWS

2.3.1
Geschichtlicher Abriss

Am 10. November 1983 wurde Microsoft Windows als Erweiterung von MS-DOS mit einer grafischen Benutzeroberfläche angekündigt. Nach einiger Verspätung erfolgte am 20. November 1985 die Lancierung, ein Jahr nach dem Macintosh. Das Windows-1.0-Package beinhaltete MS-DOS, Calendar, Cardfile, Notepad, Terminal, Calculator, Clock, Reversi, Control Panel, PIF (Program Information File) Editor, Print Spooler, Clipboard, RAM-Drive, Windows Write und Windows Paint. Die Benutzeroberfläche war ein Misserfolg und nahezu unbenutzbar, so dass Microsoft im Herbst 1987 die Folgeversion auf den Markt brachte.

In Windows 2.0 konnten sich nun endlich Fenster überlappen, viele Icons waren hinzugefügt worden, das System wurde immer besser benutzbar.

Am 22. Mai 1990 wurde Windows 3.0 veröffentlicht. Die verbesserte Speicherverwaltung und komplett überarbeitete Benutzerschnittstelle zog nun endlich auch unabhängige Softwarehersteller an. Die Fülle an Programmen verhalf Microsoft über 10 Millionen Pakete zu verkaufen, womit Windows 3.0 das meistverkaufte System in der Geschichte der grafischen Betriebssysteme wurde.

Windows 3.1 und 3.11 folgten kurz hintereinander im April 1992, ebenso Windows for Workgroups 3.1. Die neuen Versionen verbesserten das GUI beträchtlich und erweiterten die Netzwerkfähigkeit. Unter anderem erblickten Microsoft Mail und Schedule das Licht der Welt – zwei Programme, die im LAN (Local Area Network – lokaler Netzverbund) eingesetzt werden konnten.

Im Jahre 1994 stieg Microsoft in den Markt der High-End-Plattformen ein und entwickelte Windows NT 3.1. Das neue System basierte auf einem UNIX-Kernel, dessen Benutzeroberfläche an Windows 3.11 angepasst war. Einen Monat später wurde Windows NT 3.5 freigegeben, das einige Speicherprobleme behob und die Geschwindigkeit verbesserte. Windows NT 4.0 wurde ebenfalls 1994 veröffentlicht. Es war das erste Windows-System, das durchgängig objektorientiert programmiert war.

1995 kam Windows 95 als Nachfolger des Windows 3.11 auf den Markt. Es war ein Zwitter aus alten 16- und neuen 32-Bit-Routinen, hatte volles pre-emptives Multitasking, ein überarbeitetes File-System, eingebaute Netzwerkkomponenten, ein neu gestaltetes User-Interface und vieles mehr.

In dieser Zeit wurde Windows als Windows CE für Handhelds und andere Kleingeräte portiert.

Windows 98 kam als Nachfolger zu Windows 95 im Juni 1998 auf den Markt. Der integrierte Web Browser gab selbst dem Desktop ein Browser-ähnliches Aussehen, was dem Unternehmen einen großen Rechtsstreit bescherte. Multiple Display Support ermöglichte das Ansteuern von bis zu acht externen Monitoren und die neueste Hardwaretechnologie wurde integriert: DVD, Firewire, USB und AGP.

Windows NT 5.0 wurde als Nachfolger zu NT 4.0 angekündigt und versprach, Interface und Treiberausstattung aus Windows 98 zu übernehmen. Im Februar 2000 wurde Windows NT 5.0 als Windows 2000 verkauft. Das System enthielt eine imposante Vielfalt an Möglichkeiten mit hoher Sicherheit und Stabilität. Die Einbindung verschiedenster Netztechnologien machten Windows 2000 zu einem extrem leistungsfähigen und flexiblen System.

Ende 2000 führte Microsoft die alte DOS-abhängige Produktlinie mit Windows Me (Millenium Edition) fort. Das Me-System stellte vor allem multimediale Fähigkeiten zur Verfügung und war klar auf den Home-User-Markt ausgerichtet. Allerdings blieb es an Stabilität und Netzwerkfähigkeiten weit hinter Windows 2000 zurück. Microsoft entschied sich, die zwei Produktlinien zusammenzuführen.

Am 25. Oktober 2001 wurde Windows XP offiziell eingeführt. In seinem Innern steht der 32-Bit-Kernel und die Treiber des Windows NT respektive 2000. Doch noch immer kann der User alte DOS- und Windows-Programme emulieren lassen. Das Benutzerinterface wurde wieder überarbeitet und etliche neue Funktionen integriert.

Diese Arbeit wird sich auf Windows 2000 beschränken.

2.3.2
Windows 2000

Windows 2000 ist ein modernes 32-Bit-Betriebssystem mit Multiprocessing- und Multithreadingfähigkeit. Es vereint die Sicherheit und Stabilität von Windows NT, das nicht auf MS-DOS, sondern einer neuen Betriebssystemtechnik aufbaute (NT bedeutet New Technology), und die Benutzerfreundlichkeit von Windows 98. Jeder Benutzer hat seine eigene Arbeitsumgebung, die er selbst gestalten kann und die nach jeder neuen Anmeldung wieder zur Verfügung steht. Windows 2000 ist kein Multiuser-Betriebssystem, auf dem mehrere Benutzer gleichzeitig Programme ablaufen lassen könnten. Windows 2000 stellt jedoch File-Server- und Web-Server-Dienste zur Verfügung.

Microsoft integrierte den Internet-Explorer 5.0 und erweiterte sein altes File-System NTFS (New Technology File System) durch eine Echtzeitverschlüsselung (Encrypting File System – EFS), automatische Komprimierung und Active-Directory-Dienste.

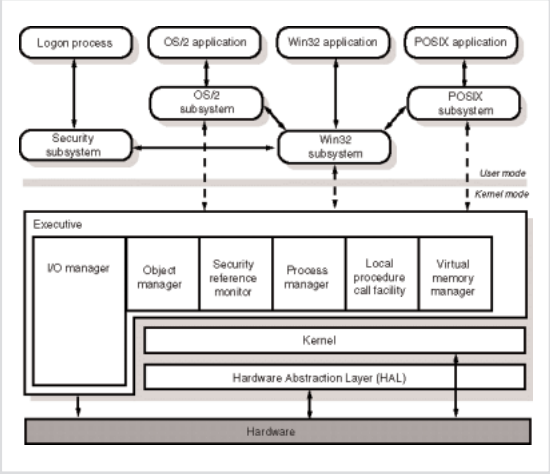
Windows 2000 ist als Windows 2000 Professional für Arbeitsplatzrechner, Windows 2000 Server für Workgroups, Windows 2000 Advanced Server für größere Abteilungen oder ganze Firmen und Windows 2000 Data Center Server, die mächtigste Version, die Multiprozessor-Rechner mit bis zu 16 Prozessoren unterstützt, erhältlich. Die Server enthalten File-Server-Lösungen für Windows, DOS und Macintosh, Internet-Server (Internet Information Server – IIS) und vieles mehr.

2.3.2.1
Systemarchitektur

Windows 2000 ist stark modular aufgebaut. Das System lässt sich grob in zwei Bereiche einteilen: den Benutzer- und den Kernel-Modus.

Innerhalb dieser Modi liegen die verschiedenen Systemkomponenten angeordnet, quasi als Schichten zwischen der Hardware und den Anwendungsprogrammen, die der Benutzer ablaufen lässt. Ihre Bedeutung wird nun einzeln erklärt.

Einen vereinfachten Überblick gibt folgende Grafik:



2.3.2.1.1
Hardware-Abstraction-Layer (HAL)

Um möglichst hardwareunabhängig zu sein, ist diese unterste Schicht für die Abstraktion der eingesetzten Hardware zuständig. Hier wird direkt auf den verwendeten Prozessor und andere Besonderheiten eingegangen.

Leider ist es möglich, dass Gerätetreiber den HAL umgehen und direkt auf die Hardware zugreifen. In einem solchen Fall ist die Systemkonfiguration entscheidend für die Betriebssicherheit – ein Zustand, der zu vielen Abstürzen führen kann und deshalb von Treiberprogrammierern vermieden werden sollte.

2.3.2.1.2
Kernel

Der Mikrokernel steuert den Prozessor und die elementaren Abläufe bei der Prozesskommunikation und ist damit ein Kernbestandteil des Betriebssystems. Hier werden die von Applikationen angeforderten Prozessorleistungen aufgeteilt (bei mehreren Prozessoren) oder so gesteuert, dass die Verteilung und Einhaltung von Prioritäten korrekt verläuft (Multitasking).

Der Kernel ist für den Benutzer unsichtbar. Er tritt nur dann sichtbar hervor, wenn der grafische Teil des Startprozesses noch nicht gestartet wurde oder bei einem Ausnahmefehler (hier erzeugt der Kernel sämtliche Blue Screens).

2.3.2.1.3
Treiber
Gerätetreiber sind spezielle Programme, die den Zugriff auf Hardwareerweiterungen erlauben. Sie werden zumeist von den Hardwareherstellern geliefert, wobei Windows für viele Standardgeräte schon Treiber mitliefert. Fehler auf dieser Ebene sind nicht durch die Schutzmechanismen des Betriebssystems abfangbar. Einfrieren oder Abstürze sind die Folge. Der Benutzer kann nur auf einen stabileren Treiber des Herstellers warten.

2.3.2.1.4
Dienste
Die oberste Schicht im geschützten Kernel-Modus bilden die ausführenden Dienste (Executive Services). Hier werden grundlegende Ein- und Ausgabeprozesse erledigt. Im Folgenden werden einige wichtige Komponenten näher erklärt.

Windows 2000 ist objektorientiert programmiert. Somit werden alle Prozesse, Threads, Sicherheitsabfragen und so weiter als Objekt behandelt. Der Objekt-Manager überwacht und verwaltet diese Objekte.

Prozess- und Thread-Objekte können mit dem Task-Manager angezeigt werden. (Jedes Programm, das startet, muss mindestens ein Prozess-Objekt verwenden.) Viele Prozesse sind jedoch selbst im Task-Manager unsichtbar. Die Kommunikation zwischen Prozessen und Applikationen geschieht über Local Procedure Calls (LPCs) in der Prozesskommunikation.

Die Überwachung und Zuteilung von verfügbarem Arbeitsspeicher zu den Programmen übernimmt der Speichermanager. Falls der physikalische Speicherplatz nicht ausreichen sollte, wird Speicher in der virtuellen Auslagerungsdatei zugewiesen. Der Speichermanager heißt deshalb auch Virtual Memory Manager (VMM).

Der I/O-Manager stellt alle Ein- und Ausgabevorgänge zur Verfügung. Hier ist auch das Dateisystem implementiert. Er bildet unter anderem die Schnittstelle zwischen Gerätetreibern von Ein-/Ausgabegeräten und den installierten Dateisystemen. Zum Beispiel ermöglicht er durch seinen modularen Aufbau das Ablaufen mehrerer Netzwerkprotokolle auf einer einzelnen Netzwerkkarte.

Die Kontrolle und Steuerung der Sicherheit übernimmt der Sicherheitsmonitor. Er prüft zum Beispiel ständig, ob ein Benutzer zu der gewünschten Aktion autorisiert ist oder führt den Anmeldeprozess bei Systemstart aus.

Hinter dem Graphical Device Interface (GDI) verbirgt sich unter anderem der Fenstermanager, der für die Erzeugung und Verwaltung der Fenster und für die Erledigung von Teilaufgaben der Druckprozesse zuständig ist. Die Platzierung im Kernel – und damit treibernah – sorgt für hohe Performance.

Weitere wichtige Bestandteile sind die Plug-&-Play-Steuerung und das Powermanagement.

Mit den Diensten endet der Kernel-Modus. Auf ihm setzt der Benutzer-Modus mit seinen Subsystemen auf.

2.3.2.1.5
Subsysteme
Die Environment Subsystems stellen APIs für ein spezielles Betriebssystem zur Verfügung. Es existieren drei Environment Subsystems: Das Win32-, das OS/2- und das POSIX-Subsystem. Obwohl es damit theoretisch möglich ist, OS/2- und POSIX-Programme laufen zu lassen, hat diese Fähigkeit nahezu keine praktische Relevanz. Weit wichtiger ist das Win32-Subsystem. Es stellt die Ausführungsumgebung für 32-Bit- (Windows 2000, Windows NT und Windows-95/98-Programme), 16-Bit- (Windows 3.1) und DOS-Applikationen zur Verfügung. Insbesondere die grafische Benutzeroberfläche wird von diesem Subsystem verwaltet.

Windows-32-Applikationen setzen direkt auf dem Subsystem auf. Ihre Speicherbereiche sind voneinander getrennt. Ein Fehler in einem Programm beeinträchtigt somit kein anderes. Die 16-Bit- wie die DOS-Applikationen werden jeweils in einem getrennten Speicherbereich in der virtuellen DOS-Maschine (VDM) ausgeführt.

Unter Windows 2000 läuft somit Software, die für MS-DOS, Windows 3.1, Windows 95, 98 und NT programmiert wurde, solange diese Software nicht direkt auf die Hardware zugreift. Dieser Zugriff ist unter Windows 2000 nicht erlaubt, um die Betriebssicherheit

zu garantieren. (Abstürze von Windows 2000 sind meist auf die schlechte Qualität der eingebauten Hardware und der damit verbundenen Treiber zurückzuführen. Billige Hardware kann hier durchaus ein Nachteil sein.)

Ein Subsystem kann sogar abstürzen, ohne dass der Kernel beeinträchtigt wird, auch wenn die strikte Trennung von Executive und Subsystemen aus Performance-Gründen seit Windows NT 4.0 leicht durchbrochen wurde.

Die Integral Subsystems stellen weitergehende Systemoperationen zur Verfügung. Die wichtigsten Vertreter sind das Security-Subsystem, das unter anderem für den Log-on-Prozess und die allgemeinen Sicherheitsüberprüfungen zuständig ist, und das Network-Subsystem, das APIs für die Netzwerkkommunikation zur Verfügung stellt.

2.3.2.2 Dateisystem

Windows 2000 unterstützt die folgenden Dateisysteme: FAT (File Allocation Table), CDFS (CD-ROM File System), NTFS (Windows NT File System) und einige Dialekte.

FAT ist das klassische DOS/Windows-Dateisystem. Es wurde erstmals mit MS-DOS eingeführt und enthält damit eine große Zahl an Altlasten, die in der heutigen Zeit stark einschränkend wirken. Die bekannte 8+3-Namenskonvention alter DOS-Systeme (8 Zeichen Dateiname plus 3 Zeichen Dateiendung) wurde mit einer überarbeiteten Version, dem FAT32, auf 255 Zeichen inklusive der Endung erweitert. Auch die maximal ansprechbare Plattenkapazität ist den heutigen Plattengrößen angepasst worden: Es lassen sich bis zu 2 Terabyte verwalten. Das FAT-System war noch nicht auf die Sicherheitsbedingungen eines modernen Betriebssystems abgestimmt, so dass ein neues File-System definiert werden musste.

Natürlich sollte ein großer Vorteil nicht unerwähnt bleiben: Das FAT-System wird von nahezu allen wichtigen Betriebssystemen unterstützt.

Das neue File-System, das die Möglichkeiten besser ausschöpfen sollte, wurde NTFS getauft. Es unterstützt Dateinamen bis 255 Zeichen, die im Unicode abgelegt werden. Es besteht keine Limitierung der Pfadlänge mehr. Die maximale Partitionsgröße liegt wie die maximale Dateigröße bei 2×10^{64} Bytes. Eine Datei ist aus so genannten Attributen (streams) aufgebaut. Damit kann Windows 2000 auch mit MacOS-Complex-Files umgehen: Die Ressource-Informationen werden in einem eigenen Stream gespeichert. Die Verzeichnis-Hierarchie (Directory) ist – ähnlich dem MacOS – in einen B-Baum gegliedert und wird neben den Dateien, diversen Log-Dateien und Verzeichnissen im Master-File-Table (MFT) gespeichert, dem Herz des Dateisystems. Diese wichtige Tabelle exisitiert mehrfach auf dem Datenträger, um die Ausfallsicherheit zu erhöhen.

Unter NTFS sind die von UNIX und MacOS X bekannten Symbolic-Links möglich, werden aber von Windows 2000 nicht benutzt. Zur Wahrung der Kompatibilität werden nach wie vor die .lnk-Verknüpfungen des Windows 98 verwendet.

Nach dem Vorbild einer Datenbank legt NTFS so genannte Transaktions-Logs an. Ein Dateibearbeitungsvorgang wird damit entweder vollständig oder gar nicht durchgeführt. Um abgebrochene Operationen nicht zu verlieren, werden sie im Log gespeichert. Der Anwender erhält allerdings keine Möglichkeit Einfluss auf die Transaktions-Logs zu nehmen.

Wird ein Festplattensektor als defekt erkannt, verschiebt NTFS die gespeicherten Daten auf einen intakten Sektor und markiert den Ursprungssektor als »bad«.

Unter Windows 2000 ist es möglich, für mehrere (zwei bis drei) Festplatten RAID-Level (Redundant Array of Independant Discs) zu definieren und sie zu einem Software-RAID zusammenzuschalten.

Ferner ist im NTFS eine Echtzeit-Lauflängen-Kompression direkt ins Dateisystem eingebaut. Sie kann im Eintrag »Eigenschaften« im Kontext-Menü datei- beziehungsweise verzeichnisspezifisch ein- oder ausgeschaltet werden. Diese Technologie ist besonders wirksam bei reinen Text-Files, also beispielsweise Log-Files, und unkomprimierten Bilddaten.

2.4 DATENAUSTAUSCH ZWISCHEN DEN WELTEN

Aus dem eben Dargestellten ergibt sich zwangsläufig die Frage, ob und welche Hürden beim Datenaustausch zwischen den Betriebssystemen zu nehmen sind. Hier soll nicht darüber philosophiert werden, welches System das bessere und welches das schlechtere sei, sondern auf nachprüfbare technologische Unterschiede eingegangen werden, die einen Datenaustausch erschweren können. Um den Umfang des Kapitels einzuschränken, werden hauptsächlich dieser Arbeit dienliche Fakten erörtert, solche also, die für den Austausch von Textdateien bedeutend sind.

Wie in Kapitel 2.2.2.2 beschrieben, setzte das klassische MacOS als einziges Betriebssystem auf die Multi-Fork-Technologie.

Der Vorteil dieses Modells war, dass Programme aus einem einzigen File bestehen können. Sie lassen sich einfach per Drag & Drop installieren und deinstallieren oder von einem beweglichen Datenträger starten. Ist bei einem Systemabsturz ein Programm beschädigt, kann es durch einfaches Austauschen wieder funktionstüchtig gemacht werden; das lästige Suchen nach der zerstörten Datei in Programm-Unterordnern entfällt.

Aus dem ständig wachsenden Anspruch an Software, der steigenden Komplexität und der beständigen Entwicklung resultieren mittlerweile allerdings Programme, die eine Unmenge verschiedener Plug-ins, Erweiterungen, Filter und Treiber benötigen und nur mehr schlecht in den Ressourcen-Zweig eingefügt werden können. Das System der complex files wird mit wachsender Komplexität zunehmend fraglich.

Der größte Nachteil ist in der mangelnden Kompatibilität zu anderen Betriebssystemen zu sehen. Speichert ein Programm wichtige Informationen im Ressourcen-Zweig einer Datei, gehen diese auf einem Windows- oder UNIX-System verloren, da hier nur der Daten-Zweig gespeichert wird. Die Datei wird unbrauchbar. Um beispielsweise Mac-Dateien auf einem UNIX-Server zum Download bereitzustellen oder einfach per Mail-Client verschicken zu können, müssen die Ressourcen zu den Daten geschrieben werden, damit das UNIX-System die Datei vollständig speichern kann. Dies geschieht über die MacBinary- oder Binhex-Codierung, die mittels verschiedener Komprimierungsprogramme erzeugt und wieder rückübersetzt werden kann.

Werden also von einer Anwendung Textformatierungsbefehle im Ressourcen-Zweig abgelegt, gehen diese bei einem Systemwechsel verloren. Demzufolge sind alle Anwendungen, die im MacOS wichtige Informationen in die Resource-Fork schreiben, am besten zu meiden.

MacOS X kehrt sich mit seinem UNIX-Kernel von der Multi-Fork-Technologie ab und setzt damit auf Kompatibilität – sicherlich ein Schritt in die richtige Richtung.

Die unterschiedlichen File-Systems bergen noch eine Unzahl weiterer Hürden. Ist die DOS-Problematik der 8+3-Namenskonvention mit Windows 95 behoben worden – es lassen sich nun 255 Zeichen als Dateinamen verwenden –, ergeben sich Übernahmeprobleme auf das Classic MacOS, genauer die Dateisysteme HFS und HFS+, die jeweils nur 31 Zeichen verwalten. Die langen Windows-Namen werden auf 8+3 gekürzt oder – mit Erweiterungen wie Joilett-Volume-Access – auf 31 Zeichen abgeschnitten. Damit können wichtige Informationen über die Datei verloren gehen, da sich die Dateiendung, mit der Windows die Zuordnung zur Applikation vornimmt, am Ende des Dateinamens befindet.

In diesen Zusammenhang gehört noch die Problematik der Pfadtrennzeichen (Delimiter). Windows verwendet einen Backslash (\) in seiner Pfadhierarchie, das MacOS einen Doppelpunkt (:). Mit MacOS X hält der UNIX-konforme Slash (/) Einzug ins MacOS. Demzufolge dürfen Dateinamen im MacOS zum Beispiel Backslashes verwenden, in Windows hingegen nicht. Dort darf aber ein Doppelpunkt stehen, der im MacOS verboten ist.

Jedes Betriebssystem verlangt seine eigene Zeilenendcodierung und schreibt Daten im systemeigenen Encoding. Diese umfangreiche Problematik stellt einen Hauptbestandteil dieser Arbeit dar und wird deshalb in Kapitel 4 ausführlich dargelegt.

Kapitel 3

Kurzer Abriss über die Programme

3.1 TEXTVERARBEITUNG MS WORD [6]

Microsoft ist mit seiner Textverarbeitungs-Software MS Word in diesem Bereich der weltweite Marktführer. Charles Simonyis, gebürtiger Ungar, ist einer der Väter von Microsoft Office, das für geschätzte 250 Millionen Menschen Textverarbeitung, Tabellenkalkulation, Präsentation, Kommunikation und Organisation besorgt. Als er 1980 zum ersten Mal mit Bill Gates zusammentraf, arbeitete er damals für das berühmte Xerox PARC Labor. Dort entwickelte er mit Butler Lampson seit 1973 an »Bravo«, der ersten Textverarbeitungssoftware für einen der Vorgänger des PC, den Alto von Xerox. Das Programm übersetzte Text direkt von der Tastatur auf den Bildschirm, einschließlich Formatierungen wie fett oder kursiv gesetzten Wörtern. Da Xerox keinerlei Interesse an einer Markteinführung hatte, Simonyis sein Programm aber kommerziell nutzen wollte, verkaufte er an Gates. 1981 entwickelte er als Leiter eines kleinen Programmiererteams seine Bravo-Idee weiter, die dann 1983 unter dem Namen »Word« auf den Markt kam.

20 Jahre später ist aus dem damaligen Word, das inklusive des Betriebssystems auf einer einseitigen Fünfeinviertelzoll-Diskette Platz fand, ein Programm mit riesigem Funktionsumfang geworden. Es korrigiert automatisch Tippfehler, schlägt Synonyme nach, ermöglicht Serienbriefe und lässt sich völlig frei auf den Benutzer konfigurieren.

3.2 DTP-SOFTWARE

1985 erschien das Programm »PageMaker« für den Macintosh. Die Firma Aldus führte mit dem Release gleichzeitig einen neuen Begriff auf dem PC-Markt ein: Desktop-Publishing (DTP).

DTP bedeutet frei übersetzt: »Publizieren vom Schreibtisch aus.« Gemeint ist das technische Herstellen von Drucksachen jeglicher Art am Computer unter Zusammenführung aller dazu benötigten Elemente (Text-Bild-Integration) und der Möglichkeit zur Ganzseitenausgabe.

DTP verfolgte im Gegensatz zu den aus der traditionellen Linie entstandenen Geräten des Computer Aided Publishing (CAP) einen anderen Ansatz. Die traditionellen Verfahren beruhten auf der getrennten Herstellung von Text und Bild, Seitenumbruch und -montage, während DTP die Integration von Text, Bild und Grafik schon beim Entwurf berücksichtigte. CAP-Stationen waren befehlsorientiert und erlaubten eine Betrachtung des Endergebnisses erst in späteren Versionen auf einem zusätzlichen Gestaltungsmonitor. DTP machte sich die neuen Errungenschaften der Graphical User Interfaces (GUIs) zunutze und war somit menü- und fenstergesteuert. Es unterstützte »What you see is what you get« (WYSIWYG) quasi »On the Fly«. Während die CAP-Systeme an die Hersteller bestimmter Geräte gebunden waren, versuchten die DTP-Programme unterschiedliche Ausgabegeräte anzusteuern und möglichst systemunabhängig arbeiten zu können. Eine wichtige Voraussetzung hierfür war die Seitenbeschreibungssprache PostScript [12]. Weitere wichtige Vorteile der DTP- gegenüber den CAP-Programmen waren ihre leichte Handhabbarkeit und der extreme Anschaffungskostenvorteil – zwei Merkmale, die bis in die heutige Zeit Bestand haben.

In den Anfängen des DTP galt eine strikte Trennung zum CAP. Ersteres wurde nur für Bürokommunikation (kleinere Broschüren, Flugblätter und so weiter) eingesetzt, während letzteres bei anspruchsvollen Satzarbeiten Anwendung fand. Im Laufe der Zeit begann jedoch eine Annäherung. Die CAP-Systeme übernahmen wichtige Elemente aus dem DTP und umgekehrt. So verschwanden durch die Einführung von WYSIWYG bei CAP-Systemen allmählich die Gestaltungsmonitore und die Bindung an bestimmte Herstellerkonfigurationen. Im DTP wurde der Funktionsumfang mehr und mehr vergrößert, so dass heutige Programme teilweise schon für hochwertigere Satzarbeiten eingesetzt werden können. Ab 1989/90 endet offiziell die strikte Trennung und DTP wurde – trotz der im Vergleich noch immer nur mittleren Satzqualität – zum festen Bestandteil des professionellen Publishing-Prozesses in den Druckereien.

Die Firma Aldus wurde 1994 von Adobe Systems Inc. übernommen. Das Ur-DTP-Programm PageMaker existiert noch heute, doch sieht Adobe seinen Platz nicht mehr als Konkurrent zum weit verbreiteten Quark XPress, das damals seine Konkurrenz weit hinter sich gelassen hatte, sondern im Bereich Bürokommunikation beziehungsweise als Einstiegssoftware[12]. Auch der Adobe FrameMaker ist keine DTP-Software im eigentlichen Sinne, sondern findet seinen Platz im Bereich komplexer technischer Dokumentationen[13], die zumeist mit starker SGML/XML-Unterstützung entstehen.

Trotz umfangreicher und bemerkenswerter Bemühungen anderer Hersteller, konnte bisher keine Software entwickelt werden, die Quark XPress hätte das Wasser reichen können: Calamus, MegaPress, VivaPress, Ventura Publisher und andere. Die CAP-Systeme sind auch heute noch durch 3B2, TextLine und andere vertreten. Die einstmals führenden Hersteller wie Berthold, Linotype, Scangraphic wurden von der Zeit überholt.

Als starkes und mächtiges Satzprogramm ist LaTEX das einzige Open-Source-Produkt auf dem Markt. Es eignet sich gut für Automatisierungen, da es eher einer Programmiersprache gleicht. Editoren mit grafischer Benutzeroberfläche helfen dem programmier-unerfahrenen Benutzer bei der Satzherstellung. Eine häufig genutzte Anwendung findet sich im Internet: Aus einer XML-Instanz werden die Daten mittels eines XSLT-Scripts ausgelesen und ins TEX-Format gewandelt. Über einen speziellen TEX2PDF-Filter, der den TEX-Code in ein funktionsfähiges PDF umwandeln kann, erzeugt beispielsweise die Deutsche Bahn AG auf ihrer Homepage die persönlichen Fahrpläne [22].

Diese Arbeit wird nur auf DTP-Programme eingehen und sich dort ausschließlich mit den beiden größten Vertretern beschäftigen: Quark XPress und Adobe InDesign.

3.2.1 Quark XPress

Die Firma Quark Inc. wurde 1981 von Tim Gill gegründet. Der Software-Entwickler erarbeitete – noch vor Apple selbst – die erste Textverarbeitung für den Apple II: Word Juggler. 1986 nahm Gill, der nur wenig Interesse an wirtschaftlichen Abläufen hatte, Fred Ebrahimi als CEO auf. Gill konzentrierte sich fortan vor allem auf die technischen Entwicklungen der Firma.

Das nächste Software-Projekt sollte eine nicht alltägliche (»very fancy« [24]) Textverarbeitung werden. Doch nachdem Gill und seine Programmierer etliche neue Funktionalitäten implementiert hatten, mauserte sich die Textverarbeitung zum DTP-Programm: Quark XPress 1.0. Die Anwendung kam 1987, ein Jahr nach dem Aldus Pagemaker, auf den Markt. Da Pagemaker die volle Unterstützung von Apple genoss, bot Quark sein Programm für $100 mehr an und verkaufte es als High-end-Produkt. Die Hochpreisstrategie ging auf und Quark erkämpfte sich langsam und stetig wichtige Marktanteile. 1988 wurde ein Distributionszentrum in Cork, Irland, gegründet, seit 1996 unterhält Quark Kunden- und Supportzentren in Dänemark, Frankreich, Deutschland, Japan und England.

1992 verkaufte Quark seine Software auch für Windows. Die Portierung dauerte fast 3 Jahre [33]. Im gleichen Jahr erschienen sowohl die Quark-XPress-Passport-Variante, eine internationale Version, die elf Sprachen unterstützte, als auch die XTension-Technologie, mit der Entwickler eigene Zusatzprogramme, so genannte XTensions, für XPress schreiben konnten. Seit Mitte der 1990er Jahre erschien eine Vielzahl neuer Produkte mit dem Ziel, medienneutrales Publishing zu ermöglichen.

Die aktuelle Version von Quark XPress ist 5.0. Quark entwickelt sein Produkt nur langsam. Während der Aldus Pagemaker 1989 schon in der Version 4.0 vorlag, erreichte Quark XPress diese Versionsnummer erst Ende 1997. Doch XPress ist nach wie vor der Standard im Bereich der DTP-Programme mit einem derzeitigen Marktanteil von geschätzten 90 Prozent [33]. Die Firma Quark ist ein Privat-Unternehmen.

Quark XPress 5.0 läuft nativ auf Windows 98 und MacOS 9.x. Alle höheren Betriebssystem-Versionen werden noch nicht direkt unterstützt. Warum Quark seine Version 5 noch nicht für MacOS X lauffähig ausliefern kann, wird heiß in der Gerüchteküche diskutiert

12 »Das ideale Seitenlayout-Programm für Unternehmen, Schulen und Selbstständige, die qualitativ hochwertige Publikationen erstellen möchten. Vorlagen, Grafiken und intuitive Design-Werkzeuge ermöglichen den schnellen Einstieg.« [1]

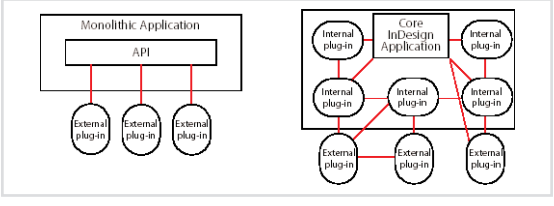
13 »Die umfassende Lösung für die plattformübergreifende Erstellung und Veröffentlichung langer, strukturierter Dokumente mit umfangreichen Inhalten. Das Programm bietet verbesserte Bucherstellungsfunktionen, darunter Befehle für ganze Bücher. Außerdem wird die Erstellung strukturierter Adobe PDF-Dateien unterstützt.« [1]

[33]. Dort ist zu lesen, dass ein Großteil der Programm-APIs Eigenentwicklungen seien, die sich nur schwer zur Carbon-Version umschreiben ließen. Des Weiteren sollen viele Entwickler das Unternehmen verlassen haben, die lange Zeit für Quark gearbeitet hatten.

3.2.2
Adobe InDesign

1994 kaufte Adobe Systems Inc. die Firma Aldus Corp. mit ihrer Publishing-Produktreihe. Darunter befand sich auch der Aldus Pagemaker – damals in der Version 5.0. Der Pagemaker war einst das führende DTP-Programm am Markt, bis Quark kurz nach dem Release von XPress ihn vom Markt verdrängte. Wie in Kapitel 3.2 schon dargelegt, definierte Adobe die Zielgruppe des Pagemaker neu. Die Code-Basis war von Aldus 1989 (das war die Version 4.0) komplett überarbeitet worden. Die Erweiterungen und Anpassungen konnten von Adobe deshalb leicht durchgeführt werden. Mit der Version 6.0 entschloss sich Adobe, die Code-Basis wieder zu erneuern. Die neue Architektur benötigte allerdings so viel Zeit in Konzeption und Durchführung, dass ein weiteres Pagemaker-Update, 6.5, parallel zu InDesign entwickelt werden musste. Nachdem Pagemaker 6.5 veröffentlicht wurde, begann das Pagemaker-Team die Hauptarbeiten an InDesign unter dem Codenamen K2.

Die Konstruktion weicht völlig von den altbekannten ab und basiert auf einem komplett objektorientierten Ansatz. Nach dem alten Ansatz können Erweiterungen (XTensions oder Plug-ins) mit der Mutterapplikation über ein fest definiertes API kommunizieren. Doch besteht für sie keine Möglichkeit untereinander Daten auszutauschen oder Funktionen gegenseitig zu nutzen. Durch die objektorientierte Struktur von InDesign steht einer solchen Kommunikation nichts im Wege. Folgende Bilder drücken diesen Sachverhalt grafisch aus:



Natürlich verbraucht eine solche Konstruktion mehr System-Ressourcen, was zu Geschwindigkeitseinbußen führt. Doch eröffnen sich damit ungeahnte, völlig neue Möglichkeiten der Erweiterbarkeit. Der größte Vorteil besteht mit Sicherheit in der besseren Aktualisierbarkeit. Die Basis-Applikation von InDesign ist nur 1,7 MBytes groß. Alles was nicht in dieser Core Application enthalten ist, wurde als Plug-in realisiert. Mit einem Bug-fix in irgendeinem Plug-in muss nur dieses und nicht die gesamte Applikation ausgetauscht oder upgedated werden. Das ermöglicht schnellere Updates mit geringerem Volumen, was Downloadzeiten spart.

Adobe sieht seine 1999 veröffentlichte Software InDesign als direkten Konkurrenten zum DTP-Klassiker Quark XPress.[14] Obwohl es ein junges Produkt ist, wartet es bereits mit einem beachtlichen Funktionsumfang auf. Adobe ist stark bemüht, seine gesamte Publishing-Linie (Photoshop, Illustrator, Acrobat und InDesign) zusammenzuführen. Die Zusammenarbeit der Adobe-Programme untereinander ist mittlerweile beachtlich. Auch ist InDesign das erste DTP-Programm das unter MacOS X lauffähig ist (wenn auch noch nicht als Cocoa-Applikation).

14 »Für professionelles Layout und Design. InDesign steigert Ihre Kreativität und Produktivität, dazu bietet es höchste Präzision.« [1]

Kapitel 4

Aufbau und Struktur von Textdateien

4.0 EINLEITUNG

In der Literatur stoßen wir häufig auf die Unterscheidung zwischen Text- (ASCII-) und Binär-Dateien. Bei dieser Betrachtung steht der Mensch im Vordergrund: Wir sprechen von einer Text- oder ASCII-Datei, wenn sie nach dem Öffnen, in beispielsweise einem Editor, einen für uns lesbaren Text ergibt; eine Binärdatei wird als nicht lesbarer Zeichensalat dargestellt.

Streng betrachtet ist diese Unterteilung unzulässig, da der Computer grundsätzlich alle Daten als eine Folge von Binärzahlen abspeichert und verarbeitet[15]. Die kleinste Informationseinheit ist dabei das Bit (abgeleitet aus Binary Digit). Acht Bits werden zu einem Byte (Bit Octet) zusammengefasst.

Was aus dieser Abfolge von Binärzahlen wie herausgelesen wird, ist vom zugreifenden Programm abhängig. Es enthält Vorschriften, Regeln wie bestimmte Bitfolgen interpretiert und bearbeitet werden. Betrachten wir es andersherum, so bestimmt das Programm maßgeblich das Aussehen, also die Struktur, der zu ihm gehörenden Datei. Diese Struktur wird Dateiformat genannt (vergleiche Kapitel 1.4).

Binärdateien enthalten zumeist ausführbaren Code, also Anwendungs- oder Systemprogramme, oder spezielle proprietäre Dateiformate (mehr dazu in Kapitel 4.3.2). Sie sind für den Benutzer nicht ohne weiteres Wissen aufzuschlüsseln.

Textdateien dagegen enthalten lesbare Zeichen, die, nach einer festen Vorschrift codiert, als Binärdaten vorliegen. Über eben diese Codierungs-Vorschriften und die verschiedenen Dateiformate soll es im Folgenden gehen.

4.1 ZEICHENSATZ, CHARSET, ENCODING [20, 26]

Einem Text liegt ein Zeichenvorrat zugrunde. Mit dem Begriff Zeichen ([abstract] character) sind nicht nur Buchstaben (Versalien, Gemeine, Ligaturen), Ziffern, Akzente und Interpunktionen gemeint, die sich je nach verwendeter Sprache mehr oder minder stark unterscheiden, sondern darüber hinaus noch etliche Sonderzeichen: mathematische, technische, chemische, botanische, genealogische, meteorologische, planetarische und viele weitere mehr.

Der Computer kann nicht direkt mit diesen abstrakten Zeichen, sondern nur mit Zahlen arbeiten. Demzufolge muss der Zeichenvorrat eines Codes ([abstract] character repertoire) definiert und jedem Zeichen eine eindeutige, ganzzahlige, nicht-negative (non-negative integer) Zahl (code number[16]) zugeordnet werden. Diese Zuordnung wird coded character set (CCS) oder character code genannt.

Die Integer-Werte des coded character sets werden in der character encoding form (CEF) eindeutig einer Binärzahl (code unit) zugeordnet. Damit ist auch die benötigte Bittiefe festgelegt (zum Beispiel 7, 8 oder 16 Bit). In den meisten Fällen existiert nur eine character encoding form (CEF) für ein coded character set (CCS), so dass der Zwischenschritt des CCS ein theoretischer ist, da jedem character direkt eine code unit zugeordnet werden kann. Die character encoding forms (CEF) sind bei der IANA[17] offiziell mit einem eindeutigen Namen (zum Beispiel Adobe-Standard-Encoding, UTF-8, ISO-8859-1, ASCII[18] und weitere) registriert.

Schließlich existiert noch ein character encoding scheme (CES). Es gibt Regeln vor, nach denen die code units in 8-Bit-Blöcke aufgeteilt werden, und definiert, ob es sich um einen Code mit fester (fixed) oder variabler (variable) Byte-Länge handelt. Diese Definitionen sind unumgänglich, da der technische Aufbau eines jeden Computers und der meisten Übertragungsprotokolle auf der Grundeinheit Byte basiert. Speicher, egal ob Hauptspeicher oder Speichermedien, werden in Bytes gemessen und adressiert, sämtliche Variablendefinitionen gängiger Programmiersprachen sind Byte-orientiert und viele Netzwerkprotokolle transferieren Daten byteweise, so zum Beispiel das TCP[19].

Obschon die heutigen Mikroprozessoren dazu im Stande wären, ergäben sich aus dem eben genannten zu viele Inkompatibilitäten und Komplikationen um eine Erweiterung der Byte-Größe auf 16, 20, 32 oder 64 Bit zu rechtfertigen. Ein weiterer wichtiger Grund, die Byte-

15 Dies ergibt sich aus der elektronischen Konstruktion. Computer-CPUs (Central Processing Units) sind aus Abertausenden von Kondensatoren aufgebaut, die als Ladungszustand nur »kein Strom« (0) und »Strom« (1) aufweisen können.

16 Es gibt eine Vielzahl weiterer Synonyme im Englischen: code position, code value, code element, code point, code set value oder schlicht code.

17 Internet Assigned Numbers Authority, http://www.iana.org/.

18 Bei der Namensreservierung sind auch Verweisnamen möglich. So ist ASCII als Synonym für den eigentlichen Namen USASCII (US American Standard Code for Information Interchange) eingetragen.

19 Transmission Control Protocol, das Basis-Übertragungsprotokoll des Internets.

Einheiten nicht zu erweitern, ist die Vervielfachung des Speicherbedarfs. Eine Speicherbelegung von 16 Bit pro Zeichen würde zu einer Verdopplung der Dateigrößen führen. Mag es für den End-User bei den heutigen Plattenkapazitäten vielleicht noch akzeptabel erscheinen, so stößt beispielsweise ein Versicherungsunternehmen, das Tera-Bytes an Daten verwalten muss, nicht nur speicherplatz-, sondern auch performancemäßig schnell an technologische Grenzen.

Die Kombination von coded character set (CCS) und character encoding scheme (CES) wird manchmal Zeichensatz (charset) genannt, obwohl die Bezeichnung den Sachverhalt nur unzureichend trifft. Auch besteht Verwechslungsgefahr, da das englische Wort font als Zeichensatz im Deutschen Verwendung findet. Man spricht in diesem Zusammenhang deshalb besser von Code(system), Encoding oder verwendet einfach den englischen Begriff Charset.

Die Gleichsetzung von Charset und Encoding existiert auch im Englischen. Der Begriff charset findet sich im HTML-Meta-Tag wieder, das den Content-Type näher definiert: `<META http-Equiv="Content-Type" CONTENT="text/html;charset=ISO-8859-1">`. XML dagegen verwendet in der XML-Deklaration die Bezeichnung Encoding: `<?xml version="1.0" encoding="ISO-8859-1" ?>`.

In der Praxis treffen wir auf eine Vielzahl verschiedener Code-Systeme. Bekannte Namen sind ASCII, ISO-8859-1, Latin-1, Unicode – doch was genau verbirgt sich dahinter?

4.1.1
7-Bit-ASCII

Der USASCII (auch standardisiert als ANSI[20] X3.4–1986 oder ISO 646) ist ein fixed 7-Bit-Encoding. Die code numbers reichen von 0 bis 127, wobei die Positionen 0 bis 31 für Steuerzeichen[21] reserviert sind. Eine Textdatei, die als 7-Bit-ASCII gespeichert wurde, kann auf nahezu jedem Betriebssystem unverändert übernommen werden.[22]

Warum eigentlich 7 Bit? Zeichensysteme sind auf dem Computer EDV-historisch gewachsene Gebilde. Bis zum Aufkommen der Personal-Computer nutzten viele Rechner noch 7 Bit lange Grundeinheiten, mit denen sich nur 128 verschiedene Zustände darstellen lassen. Der ASCII-Standard setzte sich gegenüber anderen wie beispielsweise dem EBCDIC (Extended Binary Coded Interchange Code) durch und wurde im erfolgreichen UNIX-Betriebssystem und in den aufkommenden Personal-Computern implementiert.

Mit der Aufstockung des Bytes von 7 auf 8 Bit konnten für die code numbers 128 bis 255 neue Zeichen eingeführt werden.

4.1.2
8-Bit-ASCII und ISO 8859

In der heutigen Praxis wird mit dem Wort »ASCII« leger eine Vielzahl verschiedener Codesysteme zusammengefasst, die folgende Gemeinsamkeiten aufweisen: Es sind 8-Bit-Codes, das character repertoire beinhaltet den USASCII auf den 7-Bit-ASCII-konformen Positionen 0 bis 127 als Untermenge (subset) und die code numbers reichen generell von 0 bis 255. Doch existiert eine solche Vielzahl dieser Systeme, die mehr oder minder untereinander kompatibel sind und leider teilweise gleiche oder ähnliche Namen tragen, dass hier ein großes Problempotential für die Datenübernahme aus anderen Applikationen oder anderen Betriebssystemen liegt.

Die 8-Bit-Codes im Betriebssystem MS-DOS heißen code pages (CP). Die erste wichtige CP war die CP437. Sie enthält das amerikanische character repertoire, einige griechische Buchstaben, mathematische Symbole und Sonderzeichen um einfache Rahmengerüste zu bauen.

Als Computer auch im west-europäischen Sprachraum weite Verbreitung fanden, wurde die CP437 durch den entsprechend benötigten Zeichenvorrat zur CP850 erweitert.

Die Standardisierung ISO-8859-x, die Roman Czyborra als »The ISO 8859 Alphabet Soup« [20] bezeichnet hat, beinhaltet die wichtigsten Hauptvertreter der 8-Bit-

20 American National Standards Institute, http://www.ansi.org.

21 Tabulator, Zeilenschaltung, Escape, Bell und weitere.

22 Tatsächlich gibt es Varianten des 7-Bit-ASCII. Als wirklich sicher gilt nur folgende Untermenge (subset): die Buchstaben des englischen Alphabets A bis Z und a bis z, das Leerzeichen und `! " % & ' ( ) * + , - / . : ; < = > ?`

Codes. Die Familie besteht aus 15 verschiedenen Standards, die für die unterschiedlichen Sprachsysteme des lateinischen Alphabets zum Einsatz kommen. ISO 8859-1[23] (Latin alphabet No. 1) ist beispielsweise für die Codierung west-europäischer, ISO 8859-4 (Latin alphabet No. 4) für die nord-europäischer Sprachen geeignet. Doch hat mit der Einführung des Euro ISO 8859-1 an Bedeutung verloren. Der ISO-8859-15-Standard ist als Latin alphabet No. 9 durch die Integration des Euros an dessen Stelle getreten.

Eine Auflistung der gebräuchlichsten ISO-8859-x-Encodings ist im Anhang unter Kapitel 9.1 zu finden.

Im ISO 8859-1 sind die Positionen 128 bis 159 für Kontrollzeichen reserviert, enthalten also keinerlei grafische Zeichen. Der so genannte WinLatin 1 (Windows code page 1252, der als windows-1252 von Microsoft bei der IANA registriert wurde), enthält dort aber druckbare Zeichen, so dass die Kompatibilität zum ISO 8859-1 nur unvollständig gegeben ist. Ebenso verhält es sich bei allen anderen Windows code pages (windows-1250 bis -1258).

Das Windows character set wird häufig als ANSI character set bezeichnet, obwohl es nie von ANSI standardisiert wurde; Microsofts Entwurf hatte lediglich einen ANSI-Standard als Grundlage.

Auch die coded character sets (CCS) anderer Systeme (Macintosh, UNIX, NEXTStep und andere), die programmeigenen (beispielsweise 3B2) bzw. herstellereigenen Codierungen (zum Beispiel das Adobe-Standard-Encoding für PostScript) differieren mal mehr, mal weniger voneinander. Die wenigsten jedoch weichen auf den ersten 128 Positionen vom USASCII ab.

Hieraus wird ersichtlich, dass die einfache Information, es handle sich um ASCII-Text, bei weitem nicht ausreicht um die vom Sender codierte Information verlust- bzw. störungsfrei decodieren zu können. Vorsicht ist sofort geboten, wenn ein Text über Betriebssysteme hinweg ausgetauscht wird, aber auch, wenn unter dem gleichen Betriebssystem in verschiedenen Sprachvarianten gearbeitet wurde.

4.1.3 Unicode

4.1.3.1 ISO 10 646, UCS und Unicode

ISO 10 646 ist ein internationaler Standard, der das UCS (universal character set) definiert. Es enthält ein sehr großes character repertoire, das beständig erweitert wird, und einen zugehörigen character code. Das character repertoire kann als Übermenge (superset) zu allen derzeit benutzten Code-Systemen angesehen werden.

Unicode hingegen ist ein Standard des Unicode Consortiums (http://www.unicode.org), der ein character repertoire und einen character code enthält, der voll kompatibel zur ISO-10 646-Definition ist, und ein entsprechendes character encoding scheme (CES).[24] In der Praxis werden fälschlicherweise beide Begriffe häufig gleichgesetzt.

Bei der Definition der Unicode-Encodings wurde auf Abwärtskompatibilität im USASCII-Bereich Wert gelegt. Dies zeigt folgende Tabelle für die Codierung des Buchstabens »A«:

Standard	Bits	Binär	Hex	Dec
USASCII	7	100 0001	41	65
ISO 8859-1	8	0100 0001	41	65
UCS-2	16	0000 0000 0100 0001	41	65
UCS-4	32	0000 0000 0000 0000 0000 0000 0100 0001	41	65
UTF-8	variabel	0100 0001	41	65

Im UCS-2 (unicode character set) als nativem Unicode encoding wird jede code number durch zwei aufeinander folgende Bit-Oktette – also 2 Bytes – repräsentiert. Trotz der klaren, einfachen Zuordnung, wird ihre Ineffizienz sofort augenfällig: Textdateien, die nur ISO-Latin-1-Zeichen enthalten, benötigten eigentlich nur ein Byte pro Zeichen, belegen im UCS-2 aber grundsätzlich zwei. Die UCS-4-Codierung beruht sogar auf 4 Byte pro Zeichen. Es wird im Verhältnis zu ISO-8859-1-Dateien doppelt beziehungsweise viermal so viel Speicherplatz

23 Der ISO 8859-1 ist das ursprüngliche Codesystem (dort als charset bezeichnet) für HTML.
24 Die im Februar 2000 veröffentlichte Unicode-Version 3.0 enthält 49 194 Zeichen.

nötig. Deshalb wurden andere Encoding-Varianten erdacht: zum Beispiel UTF-7-, UTF-7,5-, UTF-8- oder UTF-16-Encoding (universal text format). UTF-8 wird als Unicode-Encoding favorisiert [13], da es Daten, die fast ausschließlich USASCII- und nur wenige andere Zeichen enthalten, sehr effizient und Daten, die hauptsächlich ISO-Latin-1-Text enthalten, ausreichend effizient darstellen kann. Aus diesem Grund sei hier das UTF-8-Encoding exemplarisch näher erläutert.

4.1.3.2 UTF-8

Im UTF-8 entspricht die Belegung der Positionen 0 bis 127 dem USASCII und belegt damit nur ein Byte pro Zeichen. Alle anderen codes werden mittels einer komplexen Methode dargestellt, so dass ein Zeichen als Sequenz von zwei bis sechs Bytes erscheint. Es handelt sich also um ein Code-System mit variabler Länge.

Die binäre Repräsentation des Integer-Wertes des Zeichens wird schlicht auf die benötigte Anzahl von Bytes verteilt. Wie die folgende Tabelle darlegt, zeigt das führende Byte in den hochwertigen Stellen (high bits) die Anzahl von Bytes der Multibyte-Sequenz:

Bytes	Bits	Umsetzung
1	7	`0vvvvvvv`
2	11	`110vvvvv 10vvvvvv`
3	16	`1110vvvv 10vvvvvv 10vvvvvv`
4	21	`11110vvv 10vvvvvv 10vvvvvv 10vvvvvv`
5	26	`111110vv 10vvvvvv 10vvvvvv 10vvvvvv 10vvvvvv`
6	31	`1111110v 10vvvvvv 10vvvvvv 10vvvvvv 10vvvvvv 10vvvvvv`

Ein `v` repräsentiert hier eine Bit-Stelle, die zur Verteilung der code unit verwendet werden kann. Daraus ergeben sich gewisse Vorteile.

UTF-8 ist ein sehr kompaktes Encoding: Reine US-ASCII-Zeichen belegen nur ein Byte, die meisten anderen benötigen zwei, kein wichtiges Unicode-Zeichen mehr als drei und alle übrigen Unicode-Zeichen lassen sich mit maximal vier Bytes darstellen. Der wohl gravierendste Nachteil ist die Inkompatibilität zum ISO 8859-1, einem der wichtigsten nicht-unicode Encodings. Betrachtet man einen Latin-1-Zeichen-enthaltenden UTF-8-codierten Text in einem Programm, das nur ISO 8859-1 darstellen kann, wird der Inhalt nicht leserlich dargestellt. Es gibt noch einige weitere system- und programmiertechnische Vor- und Nachteile, die an dieser Stelle allerdings nicht dargelegt werden (weiterführend dazu empfohlen sei: [21]).

Das Encoding hat sich sehr schnell verbreitet und dürfte mittlerweile der Standard unter den Unicode-Encodings sein.

Viele Programme (vor allem im UNIX-Bereich) und einige Betriebssysteme (beispielsweise BeOS und MacOS X) unterstützen UTF-8 bereits nativ. Windows NT (und damit auch Windows 2000) kennt UTF-8 als code page 1208. Beim Classic MacOS ist UTF-8 Teil der Bibliothek »Unicode Encodings« im Ordner »Text Encodings« des Systemordners.

4.1.4 Die Problematik der Darstellung

Um Texte, und damit Zeichen, in einen Computer eingeben zu können, steht uns die Tastatur zur Verfügung. Jeder Tastendruck (egal ob direkt oder über eine Tastenkombination[25]) übergibt einen eindeutigen nummerischen Wert an den Computer, den so genannten Tastaturcode (scancode). Dieser gelangt in den Tastaturpuffer (input buffer), der vom System bzw. einer Anwendung ausgelesen werden kann.

Das Betriebssystem ermöglicht in seinen Grundeinstellungen die Wahl zwischen verschiedenen Tastaturbelegungen[26], die sogar editiert werden können. Sie definieren eindeutig das Tastaturlayout. Es existiert also eine Tabelle (eine Matrix, look-up table), die dem Tastaturcode einen Integer-Wert, den character code, zuweist. Betriebssystem und Anwendungsprogramme arbeiten grundsätzlich mit dem nativen System-Encoding. Einige

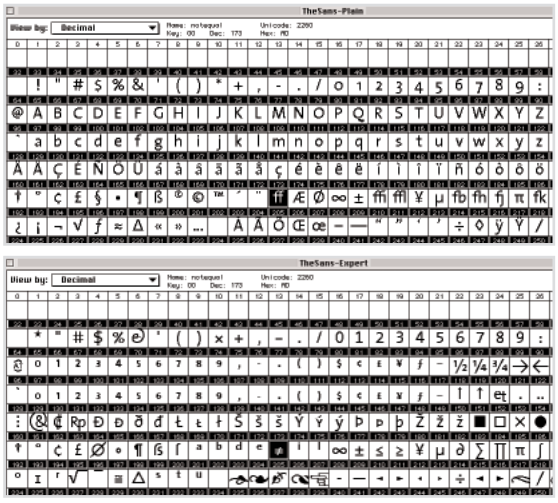
25 Gemeint sind damit die Mehrfachbelegungen, die via Alt-, Steuerung-, Apfel-Taste und so weiter genutzt werden können.

26 Beim MacOS wählt man über das Kontrollfeld »Tastatureinstellungen« die gewünschte »Tastaturbelegung«; im Windows über die Systemsteuerung »Tastatur« das »Tastaturlayout«.

Programme verwenden in ihrem Dateiformat ein komplett eigenes Encoding oder ein festgelegtes Standard-Encoding (zum Beispiel ISO-8859-1), was Plattformunabhängigkeit garantiert.

Nun kann das Zeichen entweder direkt abgespeichert oder ausgegeben werden. Beim Speichern wird der character code über das character encoding form (CEF) in eine Bitfolge übersetzt und unter Beachtung der Regeln des character encoding scheme (CES) auf das Byte-orientierte Speichermedium abgelegt.

Soll ausgegeben werden, geschieht ebenfalls ein Mapping: Der character code wird als Index für das Encoding-Array des aktuellen Fonts verwendet und verweist hier eindeutig auf die zeichnerische, visuelle Erscheinungsform des Zeichens (Glyphe), die vom System schließlich auf das Ausgabegerät transformiert wird.



Die Darstellung eines Unicode-codierten Zeichens beispielsweise bleibt somit nicht nur abhängig von der theoretischen Möglichkeit, dass ein Programm beziehungsweise ein Betriebssystem Unicode unterstützt. Es gibt derzeit nur wenige Fonts, die alle Unicode-Positionen auch mit den entsprechenden Glyphen besetzt haben. Darüber hinaus eröffnet sich noch ein weiteres Problemfeld, das auch bei Nicht-Unicode-Encodings auftritt.

Ein Font kann in diesem Zusammenhang vereinfacht als eine nummerierte Sammlung von Glyphen definiert werden. Die Nummern sollten mit den code positions der Zeichen des zugrunde liegenden Encodings korrespondieren. Leider ist dies häufig nicht der Fall. Viele Font-Designer stellen Design über Kompatibilität. Als Beispiel sei hier die Schrift-Sippe Thesis[27] genannt. Die Position U+2260[28] (Macintosh-Dec 173) ist eigentlich für das Ungleichzeichen (notequal) reserviert. Im Plain-Schnitt der Schrift findet sich dort die ff-Ligatur. Das Ungleichzeichen steht im Expert-Schnitt an der korrekten Position. Die daraus resultierende Problematik ist offensichtlich: Durch im Font falsch zugeordnete Glyphen kann eine falsche Darstellung des Inhalts im Ausgabemedium resultieren.

Die Screenshots zeigen das Layout der Thesis:

4.2 Zeilenendschaltung

Ein weiterer Unterschied zwischen Textdateien ist betriebssystembedingt und besteht in der Zeilenendschaltung. Um eine solche zu erzwingen stehen im USASCII zwei Steuerzeichen zur Verfügung: der Wagenrücklauf (Carriage Return; CR [USASCII-code 13]) und der Zeilenvorschub (Linefeed, LF [USASCII-code 10]). Die Betriebssysteme verlangen eine jeweils eigene Art und Weise, wie das Zeilenende auszusehen hat.

Zeichen	Macintosh	DOS/Windows	UNIX
Text	CR	CR und LF	LF
Dezimal	13	1310	10
Hexadezimal	0D	0D0A	0A
Binär	0000 1011	0000 1011 0000 1010	0000 1010

Hieraus ergeben sich wieder zusätzliche Probleme beim Datenhandling. Öffnet man eine Windows-Textdatei auf einem Macintosh, wird das CR als Steuerzeichen interpretiert und das LF als zusätzliches Zeichen angezeigt. Beim Öffnen einer UNIX-Textdatei wird kein

27 1994 von Luc(as) de Groot entworfen und auf der Macintosh-Plattform im Macintosh-Encoding angelegt, http://www.lucasfonts.com/.

28 Unicode-Zeichen werden häufig in der Form U+nnnn dargestellt, wobei nnnn die vierstellige hexadezimale Notation der Unicode code position eines Zeichens, das heißt ohne Bezug auf das spezielle Encoding, repräsentiert. (Die Unicode code position ist zum Beispiel im RFC 1345 [35] genauer dargelegt.) Es handelt sich hierbei um eine so genannte escape notation, auf die im Kapitel 4.3.1 genauer eingegangen wird.

Zeilenumbruch stattfinden, sondern nur die Glyphe des LF (was zumeist ein Frame, also ein kleines Rechteck ist) angezeigt.

4.3 FORMATE

Wie in Kapitel 1.4 dargestellt, unterscheidet man Dateiformate in standardisierte und nicht-standardisierte, so genannte proprietäre Formate. Im Folgenden werden Beispiele aufgeführt und somit die zwei Begriffe gegeneinander abgegrenzt.

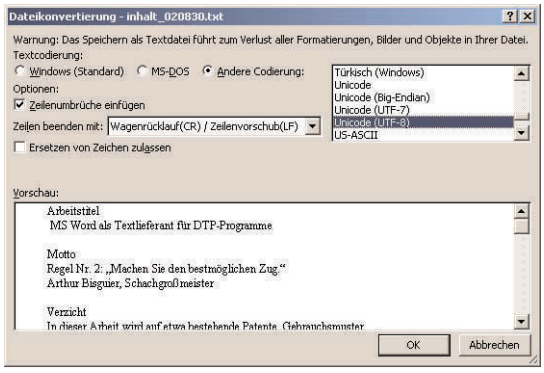
4.3.1
Standardisierte Formate

Wie in der Einleitung des Kapitels 4 schon angesprochen, gibt es in jedem Betriebssystem die Möglichkeit, Tastatureingaben ohne weitere Formatierungen in eine Datei zu schreiben. Als Codierung wird das im System eingestellte Default-Encoding verwendet (Macintosh: Macintosh, Windows: windows-1252, DOS: CP850 und so weiter). Neuere Betriebssysteme verwenden ein Unicode-basiertes Encoding (Windows 2000: UTF-16, MacOS X: UTF-8). Solche Dateien werden im Text-, ASCII- oder Nur-Text-Format gespeichert.

Zur Eingabe stehen einfache Texteditoren zur Verfügung (zum Beispiel SimpleText des MacOS oder Notepad unter Windows), die nicht mehr erlauben als ein Eingeben, Öffnen und Speichern von Textinformationen. Etwas komfortablere Editoren ermöglichen das Speichern in verschiedenen Encodings und verschiedenen Zeilenendschaltungen (beispielsweise BBEdit für Macintosh, TextPad für Windows oder GNU XEmacs, das für alle Betriebssysteme zur Verfügung steht[29]).

Auch Microsoft Word unterstützt die Ausgabe des Nur-Text-Formates. Während die Exportmöglichkeiten der älteren Versionen noch recht spärlich ausfielen, glänzt das neue Word XP für Windows mit einem ausführlichen Exportfilter. Der Benutzer kann über das System hinweg Encoding und Zeilenendschaltung ganz nach Wunsch auswählen. Leider bietet Word 2001 auf MacOS diese Möglichkeit nicht.

Nachfolgender Screenshot zeigt den Export-Dialog von Word XP:



Das Nur-Text-Format kann – unter Bezugnahme auf die in Kapitel 1.2 eingeführte Gliederung von Dokumenten – lediglich den Inhalt enthalten. Formatierung und Struktur fehlen gänzlich. Struktur und Formatierung können nur in den Text codiert werden, als lesbare Befehle. Damit bildet das Nur-Text-Format die Grundlage für verschiedene standardisierte Formate, von denen einige Beispiele nun kurz vorgestellt werden sollen.

4.3.1.1
Das Rich Text Format (RTF)

In Anbetracht der Diskussionen um die Durchweichung von C, Java, JavaScript und HTML scheint es unwahrscheinlich, dass ausgerechnet Microsoft ein offenes Standard-Format definierte, doch dem ist so. Das Rich Text Format wurde entwickelt, um formatierten Text und Grafiken problemlos zwischen unterschiedlichen Anwendungen und Betriebssystemen austauschen zu können und möglichst unabhängig vom Ausgabegerät zu sein.

Wie in Kapitel 4 erwähnt, benutzt jedes Betriebssystem beim Schreiben und Lesen von Textdateien sein eigenes Encoding, was zu Inkompatibilitäten führt. Schließlich gibt es für eine Applikation keine Möglichkeit, aus einer Nur-Text-Datei herauszulesen, mit welchem Betriebssystem sie geschrieben wurde. Folglich legt es immer sein eigenes Encoding zugrunde, was meist zu Zeichensalat bei den Zeichen, die auf Positionen größer 128 liegen, führt (wie schon in 4.1.2 dargestellt). Im RTF-

29 Auch wenn das Programm durch seine UNIX-Herkunft etwas umständlich zu bedienen ist, ermöglicht es derzeit die beste Kontrolle bezüglich Encoding und Zeilenendschaltung.

Stream gibt es eine vordefinierte Stelle, an der die Codierung vorgemerkt ist. So kann die aufrufende Applikation, die RTF-Reader genannt wird, das korrekte Encoding erkennen und die Datei entsprechend anzeigen. RTF unterstützt die folgenden Encodings: ANSI, Apple Macintosh Standard Encoding, die MS-DOS CP437 und CP850. 7-Bit-Zeichen werden direkt unterstützt, während alle 8-Bit-Zeichen als escape notations[30] hexadezimal codiert werden müssen. Damit ergibt sich – ähnlich SGML – ein reiner 7-Bit-USASCII-Stream, der beliebig transportiert und auf allen Systemen korrekt geöffnet werden kann.

In der neuen Spezifikation 1.7 wurde die Unicode-Unterstützung berücksichtigt. Der Benutzer legt direkt nach der Definition des Encodings eine zusätzliche ANSI-Codepage fest, die vom RTF-Reader benutzt wird, um die Unicode-Transformation zu erledigen. Mittels einiger Zusatzbefehle können damit sogar 2-Byte-Zeichen verwendet werden.

Die zwingende Angabe des Encodings impliziert, dass die Umsetzungslogik in der Applikation implementiert sein muss. Ein RTF-Reader muss also problemlos alle vordefinierten Encodings verstehen und zwischen ihnen transformieren können – ähnlich einem Internet-Browser. So ist es ein leichtes für den Programmierer eine Anwendung zu schreiben, die RTF lesen und korrekt auszugeben vermag – unabhängig vom Betriebssystem.

Eine RTF-Datei besteht grundlegend aus unformatiertem Text, control words, control symbols und groups.

Ein control word ist ein speziell aufgebauter Befehl, den RTF benutzt, um Steuerungsinformationen für eine Applikation oder ein Ausgabegerät zu codieren. control words haben folgenden Aufbau:

```
\zeichenfolge<delimiter>
```

Jedes control word beginnt also mit einem Backslash (\), ist gefolgt von einer Zeichenkette, die nur kleingeschriebene Alphazeichen (also: a bis z) enthalten darf, und einem delimiter, einem Trennzeichen. Die Abfolge der control words beruht, wie die Sprachdefinition von XML, auf der Backus-Naur-Form (BNF[31]). Dies ermöglicht ein relativ einfaches Transformieren zwischen den beiden Formaten.

Ein control symbol besteht aus einem Backslash gefolgt von einem nicht-alphabetischen Zeichen. `\~` beispielsweise repräsentiert einen nonbreaking space, einen nicht trennbaren Wortzwischenraum. Eine group schließlich besteht aus Text, control words oder control symbols, die in Akkoladen (`{}`) eingeschlossen sind. Sie fassen zusammengehörige Teile optisch und logisch zusammen.

Den formalen Aufbau von RTF in der Backus-Naur-Form und ausführlichen Beschreibungen detaillierter aufzuzeigen, ist für diese Arbeit nicht notwendig[32]. Es folgen deshalb schlichte Aufzählungen, um einen Überblick der Leistungsfähigkeit zu geben. Wo nötig findet sich ein tieferer Einblick in späteren Kapiteln.

Der Inhalt einer RTF-Datei folgt einem definierten Aufbau: `'{' Header DocumentArea '}'`

Im Header (Kopfbereich) der Datei stehen wichtige Meta-Informationen und grundlegende Definitionen: die RTF-Versionsnummer, das verwendete Character Set (`ANSI, PC, MAC`) sowie Hinweise auf die Unicode-Nutzung im Dokument.

Anschließend folgen Font-Informationen: Ein Font kann als Default-Font definiert werden, eine Font-Table enthält die Verweise auf alle im Dokument benutzten Fonts. Es ist sogar möglich, TrueType-Fonts in die RTF-Datei einzubetten (Fontembedding).

Um mit Sub- oder Filialdokumenten umgehen zu können, verfügt RTF über eine File Table, die Namen und URL der Unterdokumente enthält.

Als nächstes folgen Farbdefinitionen (Farbname und RGB-Wert), die im Dokument Verwendung finden, Style Sheets (Stilvorlagen/Formatvorlagen) mit allen wichtigen Funktionalitäten, List Tables (Aufzählungslisten), die sozusagen Stilvorlagen für Aufzählungen darstellen (es wird exakt festgehalten, welches Aufzählungszeichen verwendet wird, die genauen Werte der Abstände, Einrückungen und so weiter) und schließlich Track Changes (die Überarbeitungsfunktionalitäten), das sind Rückverfolgungsmöglichkeiten, welcher Autor welche Änderungen vorgenommen hat. Schließlich und endlich wer-

30 In SGML, XML und daraus abgeleiteten Sprachen wie HTML heißen die escape notations entities, zum Beispiel `ü` für das kleine ü, in der Programmiersprache PostScript wird durch `\373` im Standard-Encoding das scharfe-s (ß) repräsentiert. Manchmal werden sie auch als shading (Schattierung) bezeichnet.

31 Alle in BNF notierten Definitionen sind in dieser Arbeit in Akkoladen (`{}`) eingeschlossen. Zur BNF siehe Kapitel 9.5.

32 Weiterführend sei auf die umfangreiche Spezifikation von Microsoft verwiesen (online unter: http://www.wotsit.org, Stand: 10.10.2002).

den noch weitere Informationen über den Generator, also die Anwendung, mit der das RTF-File erzeugt wurde, festgehalten.

Die document area (der Inhaltsbereich) enthält den eigentlichen Inhalt des Dokuments. Sie beginnt mit der Information Group, die nochmals einige Meta-Informationen bereitstellt: Titel des Dokuments, Autor, Schlagwörter, Kommentar und andere Dokument-spezifische Informationen. Als nächstes folgen die Document Formatting Properties, welche die Definition der Seitenabstände, Fuß- und Kopfzonen, Fußnotenarten und so weiter enthalten.

Der nachfolgende Teil gliedert sich in folgende Sektionen auf: Section Text (Abschnitte), Paragraph Text (Text nach Absätzen unterteilt; hier findet sich auch die Tabellen-[33] und Tabulatordefinition) und schließlich Character Text (unformatierter Text). Dieser Teil kann die folgenden Elemente aufweisen: Document Variables (Variablendefinitionen, die mittels Makros angesprochen werden können), Bookmarks (Lesezeichen), Pictures, Objects, Drawing Objects (Bild- und Objektdaten können direkt in der RTF-Datei eingebunden werden, als binäre oder hexadezimale Information), Footnotes (umfangreiche Fußnoteneinträge inklusive der Darstellungsinformationen), Comments/Annotations (Kommentare), Fields (Felder, das sind Referenzen wie zum Beispiel Kapitelverweise), Index Entries (Indices wie Abkürzungsverzeichnisse oder Schlagwortregister), Table of Contents Entries (Inhaltsverzeichnisse) und zuletzt die Unterstützung fernöstlicher Sprachen, mit den dazugehörigen komplexen Definitionen.

Das RTF-Format beinhaltet den Inhalt und die Formatierung eines Dokuments, kann aber die Struktur nur unzureichend abbilden. Strukturinformationen sind lediglich implizit über die Formatierungen extrahierbar; und das teilweise nur über Heuristiken. Das Auffinden von Texten der Überschriftenhierarchie 1 beispielsweise kann dann nur über folgende Vorschriften geschehen:

1. Die Schriftart ist Caslon-Bold.
2. Die Schriftgröße beträgt 14 Punkt,
3. der Zeilenabstand 18 Punkt.
4. Der Textanfang beginnt stumpf.
5. Der Text ist linksbündig gesetzt und hat
6. am Ende keinen Punkt.

Wurde eine Absatzstilvorlage definiert, erleichtert dies das Auffinden ungemein. Damit liegt durch die Formatierung ebenso eine Strukturierung vor. Was jedoch strukturell weiter unterscheidbar sein kann – wie beispielsweise Nach- und Vorname eines Adresseintrags –, wird in der Formatierung nicht weiter differenziert.

4.3.1.2 Markup Languages

Der Terminus »Markup Language« kann als »Auszeichnungssprache« ins Deutsche übertragen werden. Eine Markup Language liefert das Regelwerk wie ein Dokument strukturell aufgebaut werden muss. Sie definiert, welche Auszeichnungen erlaubt, welche erforderlich und wie diese vom textlichen Inhalt zu unterscheiden sind und was sie bedeuten.

Markup Languages wurden aus der Idee heraus entwickelt, die Struktur von der typografischen Formatierung zu trennen. Was jedoch sind die Vorteile einer solchen Trennung?

»To get the most business value from your information you need to provide frictionless access to all of it—regardless of location, format, or source«[34] [23], schreibt das Wissensmanagement-Unternehmen EMC² in seiner Vision. Als Faustformel zusammengefasst heißt dies: Je verfügbarer und zugänglicher Informationen sind, desto größer ist ihr Nutzen. Und das gilt nicht nur im betriebswirtschaftlichen Sinne.

Der schnelle Zugriff auf Informationen – von jedem Ort, zu jeder Zeit, durch jede beliebige Person, mittels eines beliebigen Werkzeugs – ist seit langem eine Vision der IT-Welt. Damit untrennbar verbunden ist die Forderung der Nutzer, die Informationen auf ihre persönlichen Anforderungen und Wünsche hin maßgeschneidert und möglichst unverzüglich zur Verfügung gestellt zu bekommen – unabhängig vom Ausgabemedium oder bestimmten Systemen. Dazu mussten mit den vorhandenen Mitteln möglichst systemunabhängige Standards

33 Tabellen werden als Spezialfall des Absatzes behandelt.

34 »Um den größtmöglichen betriebswirtschaftlichen Nutzen aus Ihren Informationen ziehen zu können, müssen Sie einen reibungslosen Zugriff schaffen – unabhängig von Ort, Format oder Quelle.«

geschaffen werden. Die Standard Generalized Markup Language (SGML) und die Extensible Markup Language (XML) sind solche Standards, die einen großen Teil zur Realisierung dieser Vision beitragen.

Beide betrachten Daten (also den Inhalt) nicht nur als Aneinanderreihung von Zeichen, sondern als Datenobjekte (strukturierten Inhalt). Damit wird es möglich, Informationen leicht zugänglich, verarbeitbar und änderbar zu machen – und das vollkommen unabhängig von Soft- und Hardware.

Eine Information kann somit auf den unterschiedlichsten Ausgabemedien dargestellt werden, ohne dass eine Anpassung der Daten notwendig wird; ein wichtiger Vorteil in einer Zeit, in der sich die Informations- und Kommunikationstechnologie schnell weiter entwickelt. Die meisten Informationen, die heute generiert werden, müssen auch in einigen Jahren noch zur Verfügung stehen. Aber auf welchem Medium, ist nur unzureichend prognostizierbar. In den geeigneten Medien können die Daten verwendungsneutral gespeichert werden, so dass der Nutzer über vordefinierte Schnittstellen die Informationen nach seinen Bedürfnissen aufbereiten kann – er interagiert. Bei vollständiger Implementierung werden in einem Unternehmen die Informationen ein und derselben Datenquelle in den verschiedenen Abteilungen unterschiedlich (und damit auf die jeweiligen Bedürfnisse ideal zugeschnitten) dargestellt. Es entsteht also eine sehr hohe Flexibilität bei der Datenausgabe.

Die Unabhängigkeit von Hard- und Software ermöglicht ein problemloses Austauschen oder Wechseln der Hard- und Software, ohne dass eine (fehlerbelastete) Übersetzung des Datenbestands erforderlich wird, und garantiert die Langlebigkeit der Daten. Die vollkommene Trennung von Struktur und Inhalt von der typografischen Formatierung bringt auch für Autoren Vorteile. Sie können sich ausschließlich auf ihre eigentliche Aufgabe – die Generierung von Informationen – konzentrieren und damit ihre Produktivität erheblich steigern. Natürlich fällt es Autoren leichter zu schreiben, wenn sie die Endformatierung direkt vor Augen haben. Deshalb bieten geeignete Editoren die Möglichkeit, Formatierung in Form von Style-Sheets direkt nach Text-Eingabe anzuzeigen.

Es lässt sich also eine sehr hohe Nutzungseffizienz in vielen beteiligten Bereichen ausmachen.

Die nachfolgenden Darstellungen von SGML und XML beschränken sich stark und bieten lediglich einen kurzen Überblick. Weiterführendes findet sich im gut verfügbaren und umfangreichen SGML- und XML-Quellenmaterial. Für diese Arbeit wichtige technische Details werden in den entsprechenden Kapiteln aufgegriffen und kurz dargestellt.

4.3.1.2.1
Standard Generalized Markup Language (SGML)

SGML ist ein seit 1986 gültiger internationaler Standard für die Definition von Markup Languages (ISO 8879-1996). Sie zählt damit zu den Meta-Sprachen. Die Beschreibung von Dokumenten umfasst die Definition, Identifikation und Benutzung der Struktur und des Inhalts von Dokumenten. Mittels SGML kann für jede Dokumentenart (egal ob Rezepte, Telefonbücher, Lexika, E-Mails, Dokumentationen oder anderes) eine hierarchische Struktur festgelegt werden, die anschließend jedem generisch ausgezeichneten Element (Titel, Absatz, Passage und so weiter) seine vorherbestimmte Position im Dokument zuweist und festlegt, wie oft ein bestimmtes Element an welcher Stelle vorkommen darf.

Doch betrachten wir den näheren Aufbau eines SGML-Dokuments. Es besteht in der Regel aus einem Prolog und einer Dokumentinstanz. Der Prolog enthält die so genannte Document Type Definition (DTD), entweder eine vollständige oder einen Verweis auf eine extern gespeicherte. Bei der Dokumentinstanz handelt es sich um den eigentlichen Inhalt des Dokuments sowie die entsprechenden Auszeichnungen.

In einer DTD werden die Regeln für den syntaktischen Aufbau des Inhalts definiert, sie wird als Herz einer SGML-Anwendung bezeichnet und ist grundsätzlich in SGML geschrieben. Nach der Vereinbarung des Dokument-Typs (eben die Unterscheidung nach der Dokumentart), folgen die Definitionen der einzelnen Elemente und Subelemente (der Textobjekte). Die Verschachtelung

der Elemente wird hierarchisch so weit gegliedert, bis das unterste Subelement nurmehr `#PCDATA` (parsed character data), also reine Zeichenfolgen und damit reinen Inhalt enthält. Jedes Element muss einen eindeutigen Namen erhalten (generic identifier), der – im Gegensatz zu einigen Makro- oder frühen Auszeichnungssprachen von Satzsystemen – ein sprechender Name sein sollte. So wird eine Überschrift als »heading1« und nicht als »M1« bezeichnet. Je sprechender ein generic identifier ist, desto einfacher lässt sich die Dokumentinstanz auch für Menschen lesen.

Die Markup-Deklaration einer DTD für Briefe könnte – sehr vereinfacht – beispielsweise wie folgt aussehen:

```
<!ELEMENT - - brief (anrede, text)>
<!ELEMENT - - anrede (#PCDATA)>
<!ELEMENT - - text (#PCDATA)>
```

Die einzelnen Elemente finden sich in der Dokumentinstanz als so genannte Tags (Englisch für Schildchen, Etikett) wieder. Jeder Tag beginnt mit einem tag open (TAGO) und endet mit einem tag close (TAGC), das sind die Zeichen < und >. Ein Element erhält in der Instanz ein Start- (`<generic_identifier>`) und ein End-Tag (`</generic_identifier>`). Eine Beispielinstanz könnte wie folgt aussehen:

```
<brief>
<anrede>Sehr geehrte Damen und
Herren</anrede>
<text>Hiermit möchte ich Sie darauf
hinweisen, dass [...]</text>
</brief>
```

Einem Element können noch Attribute zugewiesen werden. Ein Attribut ist ein Parameter und enthält beschreibende Informationen über das Element, ohne dass diese zum Inhalt des Elements gehört. Attribute werden in der DTD deklariert und in der Dokumentinstanz immer dem Start-Tag beigefügt. Zum Aufzeigen soll nachfolgendes Beispiel genügen, in dem das autor-Tag ein Attribut namens email besitzt:

```
<autor email="tobias@wantzen.com">
Tobias Wantzen</autor>
```

Des Weiteren können in einer DTD noch Entities definiert werden. Nach ISO 8879 ist eine Entity »a collection of characters that can be referenced as a unit«[35]. Durch Entities kann also innerhalb eines SGML-Dokuments auf bestimmte Objekte verwiesen werden. Dies können ein vorher definierter Text, spezielle Symbole und Sonderzeichen, externe Dokumente oder Grafiken sein. Die wohl bekanntesten Entities sind die Umlaut-Codierungen aus HTML: `ü` für das kleine deutsche ü oder `ß` für das ß. Hieraus wird auch die Syntax deutlich. Ein Entity startet mit einem et-Zeichen (dem »Kaufmanns-und«: &), gefolgt von einer mehr oder weniger sprechenden, aber eindeutigen Abkürzung (»uuml« für »u« und »Umlaut«) und endet mit einem Semikolon »;«. Die Verwendung in der Dokumentinstanz ist denkbar einfach:

```
<zitat>Daf&uuml;r wird er mir
gradestehn m&uuml;ssen!</zitat>
```

Die Formatierung eines SGML-Dokuments übernimmt eine Style Language. Im Falle von SGML ist das die Document Style Semantics and Specification Language (DSSSL – sprich »Dissel«). Doch leistet DSSSL weit mehr als die bloße typografische Formatierung für ein Ausgabemedium. Sie kann darüber hinaus noch Transformieren (von SGML in ein anderes Format übersetzen), das Dokumentmodell abbilden (Darstellen der Dokumentstruktur) und enthält einen Abfragemechanismus (Query Language), mit dem das Durchsuchen von Dokumenten nach Inhalten, ähnlich einer Datenbankabfrage mit SQL (Structured Query Language), möglich ist.

Die bekannteste Anwendung von SGML ist HTML, die Hypertext Markup Language. HTML wird durch eine SGML-DTD definiert, welche fester Bestandteil der Internet-Browser ist. Sie ist aufgrund ihrer leichten Handhabbarkeit mittlerweile die Standardbeschreibungssprache des Internets. Doch die Vorgaben lassen für diese Sprache keine Erweiterung mit individuellen Definitionen zu, was einen Nachteil darstellt. Somit ist HTML für die Publikation auf unterschiedliche Medien nicht geeignet.

35 Sinngemäß übersetzt bedeutet dies: »Eine Anzahl von Zeichen, die auf ein Objekt verweisen.«

Des Weiteren konzentrierte man sich immer stärker auf die Kontrolle des Layouts anstatt – entsprechend dem Grundgedanken von SGML – auf die Struktur. Mit dem reinen HTML-Standard ist daher eine Trennung von Inhalt und Formatierung kaum mehr möglich.

Um diesen Nachteil zu mildern, wurde für die HTML-Definition 3.0 eine Stilsprache namens Cascading Style Sheets (CSS) entwickelt. Erst durch ihre Nutzung ist es möglich, den Inhalt vom Layout wieder besser zu trennen. CSS legt die Rahmenbedingungen fest, wie Auszeichnungen vom Browser interpretiert und dargestellt werden müssen. Sie besitzt allerdings nicht den Funktionsumfang wie DSSSL, sondern ist rein auf die typografische Formatierung ausgerichtet.

Um die Mängel und Unterschiede ein wenig deutlicher zu zeigen, dient untenstehende Tabelle. Die Gegenüberstellung zeigt noch einmal deutlich auf, dass in HTML lediglich die Formatierung des Textes durch die Tags beschrieben wird (alleine das `<p>...</p>`-Element stellt Strukturinformation dar[36]), während in SGML/XML die Tags gänzlich zur strukturellen Gliederung dienen. Damit sind weit bessere Such- und Sortiermöglichkeiten realisierbar, die bei geschickter Programmierung an den Komfort einer Datenbank heranreichen.

Der Erfolg des Internets mit der auf SGML basierenden Sprache HTML lässt schon heute erkennen, welche wirtschaftlichen und gesellschaftlichen Potentiale in SGML enthalten sind.

4.3.1.2.2

Extensible Markup Language (XML)

Angesichts der Mängel von HTML und der enormen Komplexität von SGML entstand 1996 eine weitere Sprache: XML. Wie SGML ist XML eine Meta-Sprache. Sie kann als Teilmenge von SGML verstanden werden, weil sie um einige komplexe SGML-Konstrukte erleichtert wurde. »Wie radikal die Einschnitte sind, mag man dem Umstand entnehmen, dass die formale Definition von XML auf 33 Druckseiten möglich ist. SGML benötig hierfür mehr als 500.« [28] Trotzdem bleibt XML aufwärtskompatibel zu SGML. Das im vorigen Kapitel Aufgezeigte gilt in vollem Umfang auch für XML. Die Unterschiede liegen eher im Detail. Auch in XML werden die generischen Informationen zu den verwendeten Elementen in einer DTD definiert, Inhalt, Struktur und Layout strikt getrennt und der Inhalt durch Tags strukturiert, während das Layout durch eine Style Language bestimmt wird. Durch die Konzeption ist die Nutzung

	HTML	SGML (XML[37])
Quelltext	`<p>Hermann <b>Hase</b> Frankfurter Straße 8 69476Wiesbaden geb. 3. M&auuml;rz 1927</p>`	`<person nat="d" beruf= "autor" gtag="3" gmonat="3" gjahr="1927"><vorname> Hermann</vorname><nachname> Hase</nachname><adresse> <strasse>Frankfurter Straße 8</strasse><plz>69478 </plz><ort>Wiesbaden</ort> </adresse></person>`
Darstellung im Browser[38]	Hermann Hase Frankfurter Straße 8 69476 Wiesbaden geb. 3. März 1927	Hermann Hase Frankfurter Straße 8 69476 Wiesbaden geb. 3. März 1927

36 Wenn dies auch nur unzureichend geschieht, da die strukturelle Aussage – es sei ein Absatz – nicht näher spezifiziert. Es handelt sich nämlich nicht nur um einen Absatz, sondern um Personeninformationen mit Adressangabe.

37 Das Beispiel ist, so viel sei schon voraus gegriffen, voll XML-konform.

38 Die exakte Formatierung ist natürlich abhängig von der gewählten Stilsprache und dem Browser.

einer Style Language sogar unabdingbar, während ein CSS für die Darstellung eines HTML-Dokuments nicht zwingend erforderlich ist.

Zu Recht häuften sich allerdings Stimmen, dass die Syntax zur Erstellung einer DTD zu stark SGML ähnelte. Man verlangte eine XML-konforme Syntax. Dem wurde mit XML-Schema Rechnung getragen. Aber auch hier würde eine nähere Betrachtung zu weit führen. Im Gegensatz zu SGML ist die Angabe und Definition einer DTD allerdings nicht zwingend notwendig, wenn bestimmte Kriterien erfüllt sind: Das Dokument muss »well formed« sein. Ein wichtiges Kriterium dazu ist das proper nesting; Elemente müssen danach korrekt ineinander verschachtelt sein. Der folgende Ausschnitt einer XML-Instanz wäre beispielsweise ungültig:

```
<ueberschrift1>Sonderzeichen in
<stichwort>XML</ueberschrift1>
<text>XML</stichwort> bietet für die
Sonderzeichencodierung [...]</text>
```

Unter Beachtung des proper nesting muss obige Zeile wie folgt aussehen:

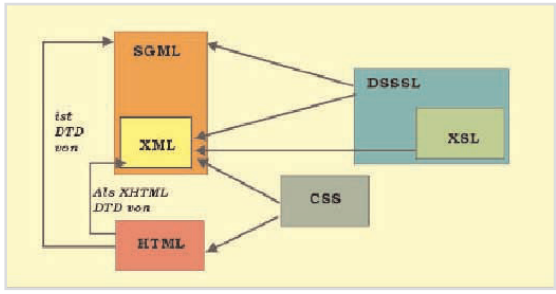
```
<ueberschrift1>Sonderzeichen in
<stichwort>XML</stichwort>
</ueberschrift1><text><stichwort>XML
</stichwort> bietet für die
Sonderzeichencodierung [...]</text>
```

Die Style Language für XML heißt Extensible Stylesheet Language (XSL) und ist stark an DSSSL orientiert. Über die reine Formatierung des Markups hinaus, kann sie ebenfalls Transformieren, das Dokumentmodell abbilden und den Dokumentinhalt untersuchen und aufbereiten. Derzeit wird XSL hauptsächlich zur Umformung von XML zu HTML als XSLT (XSL-Transformation) verwendet. Für die reine Formatierung kann auch CSS eingesetzt werden.

Die Unterschiede zwischen SGML und XML sind unter http//www.w3.org/TR/NOTE-sgml-xml genauer dargelegt. Die dort entnommene Tabelle gibt einen Überblick über die Unterschiede zwischen HTML, XML und SGML:

Eigenschaft	HTML	XML	SGML
Zahl der Auszeichnungen	feststehend	beliebig viele	beliebig viele
Komplexität	gering	mittel	hoch
Layout	pro Tag festgel., mit Attributen oder CSS	CSS, XSL oder DSSSL	DSSSL
Semantik	pro Tag festgelegt	nicht vorhanden	nicht vorhanden
Hyperlinks	einfaches `GOTO` vorgegeben	mächtiges Modell	nicht vorgegeben

Einen Gesamt-Überblick über die Zusammenhänge der hier besprochenen Sprachen zeigt die folgende Grafik:



XML wird in Zukunft eine noch bedeutendere Rolle als schon jetzt spielen – vor allem im Internet. Als Paradebeispiel sei hier das StarOffice-Paket von Sun Microsystems angeführt, das im Internet für den Privatanwender kostenlos zur Verfügung steht. Das Dateiformat von StarOffice basiert ausnahmslos auf XML, ebenso alle Präferenzdateien. Die modernen Betriebssysteme integrieren Befehle für das Verarbeiten von XML-Dateien bereits nativ.

Wichtige Markup-Sprachen, die sich aus XML entwickelt haben, sind unter anderem: XHTML, SVG und WML.

4.3.2 Proprietäre Formate

Ein öffentlich nicht zugängliches, speziell für eine Anwendung definiertes Format hat Vor- und Nachteile. Die Vorteile liegen klar auf der Hand: Die Erweiterung eines Programms um neue Funktionen kann eigenständig, ohne umständliche und langatmige Verteidigung vor diversen Gremien in das hauseigene Dateiformat integriert werden. Die Programmfunktionalität wird also – einen fähigen Programmierer vorausgesetzt – optimal vom Dateiformat abgebildet. Zudem geschieht dies nicht öffentlich, so dass kein Mitbewerber ein neues Feature einfach kopieren könnte.

Die Nachteile geben den Stoff für diese Diplomarbeit. Die notwendige Konvertierung von Daten aus einem proprietären Dateiformat in ein anderes wird ungemein erschwert. Die Arbeit kann meist nur von Spezialisten durchgeführt werden. Die fehlende öffentliche Doku-

mentation macht häufig aus einer Konvertierung eine Kryptoanalyse. Sichtbar werden die Problematiken am besten bei den Konvertierungsfiltern, die Adobe für seine Produkte schreibt, um Quark-XPress-Dokumente einlesen zu können.

Ein kurzer Seitenblick auf die Bildverarbeitungsbranche versetzt den Setzer eigentlich immer wieder in Erstaunen. Die Definition der standardisierten Bildformate (TIFF, JPEG, EPS, DCS und so weiter) ist sehr umfangreich und ermöglicht es, viele Funktionen zu nutzen. Der Datenaustausch funktioniert nahezu ausschließlich über standardisierte und nicht über proprietäre Formate. Das PDF hat, aller derzeitigen Probleme zum Trotz, das Potential auch im DTP-Bereich ein solches Standardformat zu werden.

Es soll hier keine detaillierte Auflistung der Format-Spezifikation erfolgen, sondern in allgemeiner Form über die Formate gesprochen werden. Im Besonderen kommt das jeweils verwendete Encoding zur Sprache. Tiefere Einblicke werden in nachfolgenden Kapiteln vorgenommen soweit nötig.

4.3.2.1 Adobe InDesign

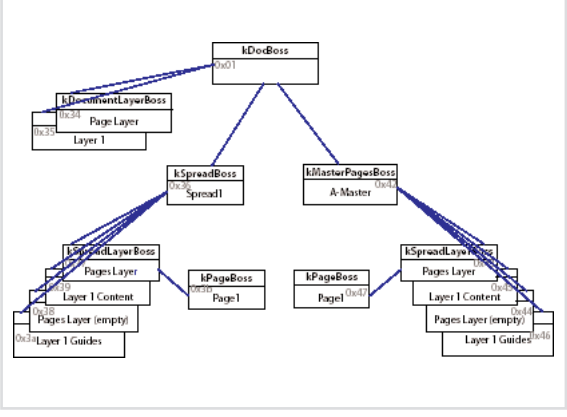
Die innere Struktur von InDesign unterscheidet sich von ihrer für den Anwender sichtbaren Erscheinung. Sie ist nicht seitenweise, sondern ebenenweise aufgebaut. Die Fläche einer Ebene (layer) ist immer so groß, dass alle im Dokument angelegten Montageflächen (spreads) darauf Platz finden. Ein Objekt wird grundsätzlich auf einer Ebene platziert und erhält dort seine festen Koordinaten. Die Einteilung in Seiten geschieht in InDesign in einem zweiten Schritt: Es existiert eine so genannte PagesLayer, eine Ebene, die einzig für die Aufteilung der Gesamtfläche in Seiten und umgebende Montageflächen zuständig ist. Über ihr liegen sämtliche Ebenen (mit ihren Hilfsebenen), die der Benutzer anlegt. Auf welcher Seite sich ein Objekt befindet, prüft InDesign, in dem es die Objektkoordinaten mit den Koordinaten der PagesLayer vergleicht. Daraus ergibt sich die Seitenzuteilung.

Ein neues InDesign-Dokument enthält automatisch die PagesLayer und eine Ebene 1 (Layer 1) mit ihrer Hilfsebene (Guide Layer 1), da eine Ebene immer aus dem Inhalt (content) und der Hilfsebene (guides) besteht. Neue Objekte werden der Ebene 1 zugeordnet, wenn der Benutzer keine weiteren Ebenen anlegt. Hilfslinien landen auf der entsprechenden Hilfsebene. Dies gilt auch für Objekte und Hilfslinien, die auf den Musterseiten (MasterPages) angelegt werden, da Ebenen dokumentweit gelten.

Der interne logische Aufbau eines InDesign-Dokuments kann informationstechnisch als Baum mit drei Unterkategorien verstanden werden: DocumentLayers (die Hauptkategorie für alle Ebenen), Spreads (die Hauptkategorie für alle Spreads, also Montageflächen) und MasterPages (die Musterseiten).

Spreads und MasterPages sind im Aufbau analog. Sie unterteilen sich weiter in SpreadLayers und Pages. In DocumentLayers werden lediglich die Ebenen verwaltet.

In einem einseitigen leeren Dokument enthält DocumentLayer automatisch den PageLayer und den Layer 1, in Spreads ist Spread 1 angelegt, in MasterPages A-Master und Pages enthält Page 1.



Adobe InDesign nutzt nicht das PDF als internes Dateiformat[39]. Ein InDesign-Dokument muss Informationen enthalten, die in einem PDF nichts zu suchen haben, weshalb Adobe sich gegen PDF als Format entschieden hat.

39 Die wenigen Informationen über das InDesign-Dateiformat sind dem InDesign-SDK-2.0 [11] entnommen. Weitere Informationsquellen sind als Nicht-Entwickler nicht zu erhalten.

Die physikalische Datei-Struktur eines InDesign-Dokuments gleicht einer Datenbank. Jedes Objekt ist wie ein Datenbankeintrag abgelegt: das Objekt selbst, Informationen über das Objekt, wo sich das Objekt befindet und wie es formatiert beziehungsweise bearbeitet ist.

Teile der Datei sind XML codiert. Ein Auszug dessen findet sich im Kapitel 9.3. Als Datei-Encoding nutzt Adobe ein Unicode-Encoding, das von allen modernen Betriebssystemen nativ unterstützt wird. Damit entfällt bei der Portierung einer Datei von einem Betriebssystem zum anderen Konvertierungsarbeit[40].

4.3.2.2 Quark XPress

Über das Quark-XPress-Dateiformat etwas zu erfahren, ist nahezu unmöglich. Im Internet existieren keinerlei Quellen, an das SDK[41] ist ohne Anmeldung und Bezahlung nicht heranzukommen. Bleibt nur die Analyse mit einem Hex-Editor[42]. So kann wenigstens das verwendete Encoding und die Sonderzeichencodierung kurz beleuchtet werden.

Auf dem Macintosh benutzt XPress das Macintosh-, auf dem Windows-System das windows-1252-Encoding. Damit fällt bei jedem Plattformwechsel Konvertierungsarbeit an. Öffnet man eine XPress-Datei mit einer XPress-Version auf einem anderen System, erscheint ein Fenster, das auf die Konvertierung hinweist.

Leider muss der strukturelle und logische Aufbau der XPress-Datei im Dunkeln bleiben.

4.3.2.3 Microsoft Word[43]

Word ist eine OLE-2.0-Applikation (Object Linking and Embedding). OLE ist ein von Microsoft entwickelter Standard (ein API-Set), der größere Flexibilität und Austauschbarkeit von Dateien zum Ziel hatte. Streng OLE-konforme Programmierung ermöglicht, dass beispielsweise ein Excel-Dokument in einem Word-Dokument platziert werden kann. Um das Excel-Dokument zu bearbeiten, kann der Nutzer mit einem einfachen Doppelklick innerhalb von Word die Excel-Funktionalitäten starten, die benötigt werden. Das UI (User Interface – die Benutzeroberfläche) verhält sich also wie eine einzige Applikation. Nach Beenden der Änderungen, werden die Excel-Funktionalitäten beendet, zurück bleibt das reine Word-Programm.

In der OLE-2.0-Definition existiert eine docfile API, die Dateioperationen OLE-konform abwickelt. Ein Word-Binary-Dokument ist ein so genanntes docfile. Daten werden in streams mit Hilfe von OLE 2.0 in das docfile geschrieben. Diese streams werden in der Datei als verlinkte Listen von Datenblöcken gespeichert. Die Daten können über die Open-APIs des Betriebssystems nicht korrekt ausgelesen und geschrieben werden. Um Daten in einem Word binary file anzusprechen, muss die Datei mittels der OLE 2.0 docfile API geöffnet und bearbeitet werden.

Während Microsoft in sein Betriebssystem Windows OLE direkt integriert hat, musste für den Macintosh eine entsprechende Erweiterung entwickelt werden. Installiert ein Benutzer Microsoft Word auf seinem Macintosh-Rechner, werden automatisch die OLE-Librarys in den Systemerweiterungen installiert. Ohne diese Erweiterungen ist kein korrekter Zugriff auf eine Word-Datei möglich.

Die neueren Versionen von Word – Word XP für Windows und Word X für MacOS X – basieren auf dem Dateiformat 8.0.

Ein Word-Docfile besteht aus einem main stream, einem summary information stream, einem table stream, einem data stream und keinem oder mehreren object streams, in denen verlinkte OLE-Objekte (wie beispielsweise Grafiken und andere Fremdobjekte) im Binärformat abgelegt werden.

Der main stream enthält den eigentlichen Inhalt des Dokuments. Der summary information stream beinhaltet Metainformationen über das gesamte Dokument, während der table stream sämtliche Tabellen fasst, die für die Struktur des Dokuments wichtig sind. Der data stream enthält wichtige Zusatzinformationen, die an den Inhalt im main stream gekoppelt sind. Damit sind also die

40 Im Anhang unter Kapitel 9.2 wurde die Codierung von Sonderfunktionszeichen in den einzelnen Programmen exemplarisch gegenüber gestellt.

41 Software Developer Kit. Es enthält alle notwendigen Bibliotheken und APIs (Application Programming Interfaces), um Erweiterungen für die entsprechende Software schreiben zu können.

42 Ein Hex-Editor stellt den Inhalt von Dateien in binärer oder hexadezimaler Darstellung dar, greift also direkt, ohne Interpretation auf die enthaltenen Daten zu. Für den Programmierer ist er ein unerlässliches Werkzeug.

43 Das Word-Dateiformat ist gut dokumentiert und zugänglich. Eine entsprechende Spezifikation kann aus dem Internet herunter geladen werden (online unter: http://www.wotsit.org, Stand: 10.10.2002).

reinen Inhalte von ihren Attributen getrennt verwaltet. Diese Technik bringt entscheidende Geschwindigkeitsvorteile bei der Bearbeitung von Daten. Nur ein Beispiel: Die Information, welche Zeichen im gesamten Word-Dokument der Stilvorlage »Normal« zugeordnet sind, muss nicht erst mühsam zusammengesucht werden. Vielmehr liegt sie im docfile schon vor: Eine Tabelle enthält alle Zeichenpositionen, die zur Stilvorlage »Normal« gehören. Im Word XP für Windows kann der Benutzer mit der rechten Maustaste auf einen beliebigen Text klicken und die Funktion »Zeichen mit gleicher Formatierung markieren« wählen. Daraufhin markiert Word XP im gesamten Dokument sämtliche Vorkommen der angewählten Formatierung, ohne merkliche Verzögerung. Solche Funktionalitäten sind erst durch die erwähnte Auftrennung von Inhalt und Attributen zu realisieren.

Textinformationen werden im eigenen XCHAR-Encoding (eXtended CHARacter set) abgelegt. Microsoft definiert character als Glyphe, egal ob es sich um einen single- oder multi-byte character handelt. Der character code basiert auf einem 16 bit Integer und korrespondiert zum Unicode character code der Glyphe. Weiterhin existieren eine Vielzahl von Sonderbefehlszeichen in Word, die für bestimmte Steuerungsaufgaben verwendet werden (siehe Kapitel 9.2). Damit entfällt bei einem Plattformwechsel Konvertierungsarbeit.

In Word gibt es die Möglichkeit der Schnellspeicherung von Dateien. Dabei werden Neuerungen und Änderungen beim Speichervorgang lediglich ans Ende der Datei angehängt; man könnte die Datei dann mit einem Schreibblock vergleichen, in den der Autor kreuz und quer hinein geschrieben hat. Damit wächst nicht nur die Dateigröße, sondern auch ihre Komplexität. Aus diesem Grund heißt eine solche Datei aus programmiertechnischer Sicht auch complex file. Bei erneuter Speicherung mit Hilfe der »Speichern unter«-Funktion, wird die Datei neu geordnet und von Word wieder »sauber« (als nicht complex file) abgelegt.

Kapitel 5

Der Testlauf

5.0 EINLEITUNG

Aufgrund der soeben dargelegten Struktur von Word-Dateien wurden die verschiedenen Transport-Möglichkeiten mit einem vordefinierten Standardtext ausgetestet, um eine bessere Vergleichbarkeit zu erreichen. Das folgende Kapitel legt kurz dar, wie diese Testdatei entstand und welche Kriterien ihr zugrunde gelegt wurden.

5.1 DIE TESTDATEI

Ziel war es, eine realistische Textmenge zu betrachten, die die zahlreichen Funktionen von Word mehrfach nutzt. Ein wissenschaftlicher Text ist wegen seiner starken Strukturierung und der Verwendung von Fußnoten, Tabellen und Grafiken optimal geeignet. Die Wahl fiel auf die Diplomarbeit von Gabriele Otterstetter, die im Internet frei verfügbar ist [31], aus rechtlichen Gründen hier allerdings nicht zitiert wird.

Die Datei enthält 268 102 Zeichen, das entspricht 149 Seiten zu je 1800 Zeichen. Der Text ist in vier Hierarchieebenen gegliedert und enthält 18 platzierte Bilddateien, die in die Datei integriert sind. Zusätzlich wurde er mit Indexeinträgen ausgestattet. Als Word-doc-File beträgt die Dateigröße 1,04 MB.

Um einen schnelleren Überblick über den Konvertierungserfolg zu erhalten, sind der Testdatei einige Seiten vorangestellt worden. Sie enthalten in kurzer Folge alle zu untersuchenden Strukturen und Elemente. Mit den dann folgenden Seiten wird die Tauglichkeit des jeweiligen Konvertierungsweges für größere Textmengen getestet. Deshalb – und aus urheberrechtlichen Gründen – werden in diesem Kapitel nur die ersten beiden Seiten besprochen und im Anhang unter Kapitel 9.5 abgebildet.

Der Export aus Word in die unterschiedlichen Formate wurde allein mit Word XP auf dem Windows-System vorgenommen, um damit die neuesten Format-Versionen (beispielsweise für das RTF) zu erhalten.

5.2 DAS UNTERSUCHUNGSPROTOKOLL

Um die Konvertierungswege aus Word in die DTP-Programme hinreichend auf die Probe zu stellen, wurde eine Vielzahl von Funktionen benutzt, die teilweise automatisch von Word verwaltet werden, teilweise aber auch absichtlich von Hand angewendet wurden.

Die Auswertungen der Versuchsergebnisse wurden in einem Protokoll festgehalten. Dabei wurde jedes untersuchte Merkmal von 0 bis 2 skaliert, wobei 0 »nicht«, 1 »ausreichend« und 2 »sehr gut konvertiert« bedeutet, und bei Bedarf das Ergebnis kurz kommentiert. Sämtliche Protokolle sind im Anhang unter Kapitel 9.7 zu finden.

Unter Kapitel 9.7.3 findet sich ein Vergleich der kumulierten Punktzahlen. Die Ergebnisdiskussionen in Kapitel 6 und 7 geschehen allerdings gänzlich qualitativ, da die Punktzahlen nur zur Orientierung und nicht zur statistischen Auswertung geeignet sind. Doch geben sie den in den Auswertungen beschriebenen Trend sehr deutlich wieder.

5.3 DAS KONVERTIERUNGSZIEL

Das Ziel der Konvertierung soll eine möglichst maximale Übertragung der getesteten Word-Funktionalitäten in das DTP-Programm sein, damit eine Automation erreicht werden kann.

Für die Merkmale und Merkmalsgruppen bedeutet dies im Einzelnen:

– *Zeichen- und absatzorientierte Formatvorlagen:*
  In ein leeres Dokument importiert muss die Stilvorlage korrekt angelegt werden: richtiger Name, korrekte Zuweisung bei den betreffenden Textstellen, korrekte Formatierung wie in Word.
  In ein Dokument mit vorgefertigtem Layout importiert müssen folgende Kriterien erfüllt werden: richtiger Name, korrekte Zuweisung bei den betreffenden Textstellen und keinerlei Übernahme von Formatierung aus Word.

- *Verschlagwortung durch Indices:* Die Verschlagwortung muss im Text erhalten bleiben und in der DTP-Anwendung direkt der entsprechenden Funktion zugeordnet werden. Die Generierung eines Indexverzeichnisses muss anschließend sofort möglich sein.
- *Querverweise (auf Kapitel, Fußnoten):* Querverweise müssen zumindest in feste Zeichen umgewandelt sein, wenn die DTP-Applikation keine dynamischen Querverweise unterstützt.
- *Fußnoten und -ziffern:* Den Fußnoten und ihren Ziffen müssen korrekt die entsprechenden Stilvorlagen zugeordnet sein. Da keine DTP-Applikation Fußnoten automatisch verwalten kann, ist eine korrekte Umwandlung in feste Zeichen zwingend notwendig. Optimalerweise stehen die Fußnoten dann am Ende des Textes.
- *Hyperlinks:* Ähnlich der Indices müssen Hyperlinks im Text erhalten bleiben und in der DTP-Applikation direkt der entsprechenden Funktion zugeordnet werden.
- *Platzierte Bilder:* Platzierte Bilder sollten so übernommen werden, wie sie im Word platziert wurden.
- *Bildbeschriftung und -nummerierung:* In ein leeres Dokument importiert muss die Stilvorlage korrekt angelegt werden und zugewiesen sein. Die automatische Vergabe der Bildnummer muss zumindest in feste Zeichen gewandelt werden.
- *Automatische Kapitelnummerierung:* Da keine DTP-Applikation automatische Kapitelnummerierung unterstützt, ist eine korrekte Umwandlung in feste Zeichen zwingend notwendig.
- *Automatische Aufzählungen:* Da keine DTP-Applikation automatische Aufzählungen unterstützt, ist eine korrekte Umwandlung in feste Zeichen zwingend notwendig.
- *Tabelle mit Formatierung (Umsetzung in Tabelle, Zellenformatierung und so weiter):* Da beide untersuchten Programme Tabellenfunktionen anbieten, muss eine Tabelle mit ihrer gesamten typografischen Erscheinung sauber und vollständig importiert werden und optimalerweise direkt im Textfluss an der betreffenden Stelle platziert sein.
- *Zusätzliche Auszeichnungen per Hand (Bold, Kursiv und so weiter):* Die zusätzlichen Auszeichnungen sollten als Funktion vollständig erhalten bleiben, damit in der DTP-Applikation nach ihnen gesucht und entsprechend ersetzt werden kann.
- *Sonderzeichen (weicher Umbruch, weiche Trennung und so weiter):* Die Sonderzeichen müssen korrekt umgesetzt werden.
- *Gesamter Text umgesetzt:* Der gesamte Inhalt der Testdatei muss importiert worden sein.

Kapitel 6

Von Word in DTP

Die Möglichkeiten, die Inhalte eines Word-Files in eine DTP-Software zu bringen, lassen sich in zwei große Bereiche gliedern: Kommunikation über Routinen des Betriebssystems und Kommunikation über Datenformate.

Zu ersteren gehören die zwei Mechanismen Copy & Paste und Drag & Drop. Zu letzteren das Word-, Nur-Text-, RTF-Format sowie XML und XPress-Marken respektive Tagged Text.

Als Ziel gilt es, möglichst viel Funktionalität des Word-Files automatisch übernehmen zu können.

Das Kapitel wird die Versuchsergebnisse beschreibend erläutern und nur auf die wichtigsten Problematiken eingehen. Die exakten Daten sind in Form der Protokolle im Anhang unter 9.7 abgedruckt.

Die Betrachtung wird sich auf Quark XPress 5.0 und InDesign 2.0 beschränken. Da zum Zeitpunkt dieser Arbeit nur InDesign 2.0 in der Carbon-Umgebung von MacOS X lauffähig ist und Quark XPress lediglich im Classic-Modus, kann ein Vergleich der beiden Programme unter OS X zu keinen klaren Aussagen führen. So stellen die Abschnitte über MacOS X nur die Möglichkeiten dar und keine Untersuchung.

Demzufolge werden die Betriebssysteme MacOS 9.2.1 und Windows 2000 betrachtet. Als Word-Versionen kommt unter MacOS Word 2001 und unter Windows Word XP zum Einsatz.

6.1 COPY & PASTE

6.1.1 Einleitung

Copy & Paste funktioniert über die so genannte Zwischenablage[44] (Clipboard oder Pasteboard). Es soll dem Benutzer ermöglichen, Daten innerhalb eines Dokuments, zwischen verschiedenen Dokumenten einer Anwendung oder zwischen Dokumenten zweier Anwendungen auszutauschen. Dem Programmierer stehen Standard-Klassen und -Funktionen des Betriebssystems zur Verfügung um Copy & Paste sehr einfach in seine Software integrieren zu können. Der User kann die Befehle per Tastaturcode (in Windows: Str-c [kopieren], Str-x [ausschneiden] und Str-v [einfügen] bzw. in MacOS: Apfel-c, Apfel-x und Apfel-v) oder über die Menüzeile des jeweiligen Programms nutzen.

Zusammenfassend kann festgehalten werden, dass es sich bei der Zwischenablage aus technischer Sicht um einen speziellen Speicherbereich handelt, zu dem Applikationen Schreib- und Lesezugriff haben, die Cut-, Copy- und Paste-Operationen unterstützen.

Leider gibt es keine Standardisierung, wie Daten auszusehen haben, die zwischen zwei Applikationen per Copy & Paste ausgetauscht werden. Die Betriebssysteme stellen die Kommunikationskanäle (über die APIs) und die Sprache (aber weder die Synatx noch den Code) für die Kommunikation als sehr einfaches Regelwerk zur Verfügung. Es werden also nur wenige Dateiformate nativ vom System unterstützt. Daraus folgt, dass ein Programmhersteller sich eigene Standards definieren muss, die damit aber meistens nur innerhalb der eigenen Produktpalette zur Anwendung kommen und nicht darüber hinaus.

6.1.2 Die Implementierung in den Betriebssystemen

6.1.2.1 Classic MacOS [28]

Beim Mac heißt die Zwischenablage aus programmiertechnischer Sicht »Scrap«[45] (manchmal auch »Scrap Desk«) und wird vom Betriebssystem, genauer dem »Scrap Manager«, verwaltet. Das Betriebssystem reserviert Speicher in jedem »application heap«[46] für den Desk Scrap und setzt einen Zeiger (handle), der mit der globalen Systemvariablen »Scraphandle« abgefragt werden kann. Beim Starten einer Applikation (das umfasst auch das Wechseln von einer Applikation in eine andere), wird der Inhalt des Scraps der vorher aktiven Applikation in den der nun aktiven hineinkopiert. Ist der Inhalt zu groß, wird er auf der Festplatte abgelegt und der Handle entsprechend auf den neuen Speicherort gesetzt.

44 Genauer ist die Zwischenablage die Visualisierung eines Speicherbereichs, der dem Copy-&-Paste-Mechanismus vom Betriebssystem zur Verfügung gestellt wird.

45 Die User-Interface-Guidelines von Apple [14] weisen ausdrücklich darauf hin, die Begriffe »Scrap« und »Pasteboard« nur im programmiertechnischen Zusammenhang zu verwenden. Dem User gegenüber sollte nur von »Clipboard« gesprochen werden.

46 Speicherbereich, der einer Applikation vom Betriebssystem zur Verfügung gestellt wird.

Das MacOS kennt drei verschiedene Formatspezifikationen, die mit dem Scrap benutzt werden können: Standard, Optional und Private Formats.

Die Standard Formats müssen von jeder Applikation unterstützt werden, die Copy & Paste nutzen will: »TEXT«, also 8-Bit-ASCII-Zeichen im Macintosh-Encoding, und »PICT«, QuickDraw-Bilddaten. Die Optional Formats können – wie der Name schon sagt – optional unterstützt werden: »snd« für einfache Sound- und »movv« für einfache Bewegtbild-Daten, »styl« für TextEdit[47]-konform formatierte Textdaten. Weiterhin hat der Programmierer mit den Private Formats die Möglichkeit für seine Applikation ein eigenes Scrap-Format zu definieren.

Der oben dargelegte System Scrap kann nur ein Objekt zur gleichen Zeit beinhalten. Alternativ kann ein Programm seinen eigenen Scrap (Private Scrap) benutzen, welcher – je nach Implementierung – unbegrenzt viele Unterscraps enthalten kann. Einige Programme nutzen dies schon (zum Beispiel Microsoft Office und die Adobe-Produktpalette).

6.1.2.2
MacOS X

Für die Carbon-Umgebung wurde der Scrap-Manager umgestaltet und auf die Bedürfnisse des MacOS X zugeschnitten. Die verschiedenen Datenformate, die darüber ausgetauscht werden können, heißen »Scrap flavours«. Sie beschreiben ein diskretes Objekt, zum Beispiel einen Text oder ein Bild. Carbon-Programme profitieren in gleichem Umfang von den erweiterten Möglichkeiten des neuen Systems wie Cocoa-Anwendungen.

Neben diversen QuickTime-kompatiblen Formaten kann der Benutzer nun auch PDF über die Zwischenablage austauschen. Das eröffnet ein weites Spektrum an standardisierten Möglichkeiten. Natürlich steht es dem Programmierer nach wie vor frei, eigene Formate für die Zwischenablage zu definieren.

Ein Objekt kann mehrfach in der Zwischenablage repräsentiert sein. Kopiert man einen Text aus einer Anwendung, so kann dieser im Hintergrund beispielsweise als Nur-Text-, Word-Format, RTF und im applikationseigenen Format in die Zwischenablage geschoben werden. Die Vorteile liegen klar auf der Hand: die Empfängerapplikation hat viele Formate zur Verfügung, aus denen sie das passende zur Datenübernahme wählen kann.

6.1.2.3
Windows

Das Procedere im Windows weicht vom MacOS in einigen Punkten ab.

Das Clipboard ist kein fester Speicherbereich wie unter MacOS 9.x, sondern lediglich ein Verweis auf die zu übertragenden Daten. Für die Übertragung muss der Datentyp festgelegt werden. Windows unterstützt neben dem ANSI-Text, Unicode-Text, TIFF, Wave, Symbolic Links und eigenen Formaten sogar das Copy & Paste ganzer Dateien und Verzeichnisse.

Eigene Formate werden vom System kategorisiert, so dass die empfangende Applikation feststellen kann, ob es sich um einen Text, ein Bild oder sonstige Daten handelt. Die Definition, welches Format übergeben wird, kann mit dem MIME-Standard (Multipurpose Internet Mail Extensions) übermittelt werden. Dieser gibt durch ein eindeutiges Informationspärchen an, um welches Format es sich handelt (zum Beispiel: »image/x-dxf« beschreibt das Standard-Vektorformat der meisten CAD-Programme DXF).

6.1.3
Versuchsergebnis

6.1.3.1
Quark XPress 5

Quark übernimmt nach einem Copy-&-Paste-Vorgang aus der Zwischenablage lediglich Nur-Text im entsprechenden System-Encoding. Sämtliche Formatierungen, Strukturen und weiteren Word-Funktionen gehen verloren. Zwar werden die Feldfunktionen (Fußnotenzahlen, Querverweise, Kapitelnummerierungen und Aufzählungszeichen) korrekt in Nur-Text-Zeichen gewandelt wiedergegeben, doch wird leider der gesamte Fußnotentext nicht übernommen.

47 Ein gutes Beispiel für den Leistungsumfang der TextEdit-APIs ist der Texteditor SimpleText von Apple, der im Lieferumfang des Betriebssystems enthalten ist.

Es ergaben sich keinerlei Unterschiede zwischen den beiden Betriebssystemen.

Damit ist dieser einfachste Weg einen Text zu transportieren recht uneffektiv. Es bleibt allerdings zu erwähnen, dass in der Vergangenheit schon so mancher Text, der sich unverständlicherweise auf keine andere Weise in XPress übernehmen ließ, auf diesem Weg erfolgreich importiert werden konnte, so dass wenigstens der Inhalt nicht erfasst werden musste. Sollte sich also eine Textdatei, trotz reger Bemühungen, einmal nicht ins DTP-Programm konvertieren lassen, ist dies die letzte Möglichkeit. Hilft auch das nicht, bleibt nur mehr ein tieferer Eingriff in die Datei über einen Hex-Editor.

6.1.3.2
InDesign 2

Das neuere Programm InDesign geht mit der Möglichkeit des Copy & Paste unter Windows 2000 weitaus besser um. Nach dem Paste-Befehl führt InDesign einen RTF-Import aus. Sämtliche Formatierungen bleiben erhalten, ebenso alle Indices, Hyperlinks und Formatzuweisungen. (Näheres zum RTF-Import siehe Kapitel 6.4.)

Unter MacOS 9 steht die veraltete Technologie des Betriebssystems im Wege. Der Datenaustausch ist noch schlechter als mit XPress 5, da InDesign versucht, so viel wie möglich zu importieren. Damit entstehen unvorhersehbare Eigenarten. Eine weiche Trennung wird beispielsweise zur harten Trennung, der gesamte Text wird in Arial BoldItalic importiert, die Fußnoten fehlen komplett. Es ist dringend anzuraten vor einem Copy & Paste aus Word den Umweg über einen Texteditor zu machen, damit alle Formatierungen verloren gehen und sämtliche Zeichen korrekt interpretiert werden.

Es ist davon auszugehen, dass die bessere Technologie unter MacOS X vergleichbare Ergebnisse zu Windows 2000 erzielt.

InDesign stellt also – auf den modernen Betriebssystemen – mit Copy & Paste eine effektive und wichtige Möglichkeit zum Datenaustausch zur Verfügung. Unter MacOS 9.2.1 sollte besser darauf verzichtet werden.

6.2 DRAG & DROP

6.2.1
Einleitung

Der Prozess des Drag & Drop ist wie folgt: Man transferiert ein Objekt von einer in die andere Anwendung, in dem man es mit der Maus erfasst (anklicken und die Maustaste gedrückt halten), auf eine offene Datei der zweiten Anwendung zieht und dort fallen lässt.

Das Drag-&-Drop-Prinzip ist zwar schon seit längerem sowohl in Windows als auch in MacOS integriert, wird aber erst seit kurzem immer stärker von Programmierern als wichtiges Feature genutzt um Objekte zwischen Anwendungen auszutauschen. Der Mechanismus selbst wird vom Betriebssystem zur Verfügung gestellt und ist dem des Copy & Paste sehr ähnlich. Als Basis steht die gleiche Funktionalität zur Verfügung, weshalb hier zur technischen Erläuterung auf Kapitel 6.1.2 verwiesen sein soll. Zu beachten gilt jedoch, dass Drag & Drop nicht über das Clipboard ausgeführt wird, sondern seine eigenen, speziellen Speicherbereiche nutzt. Damit ist gewährleistet, dass Copy & Paste und Drag & Drop nebeneinander genutzt werden können, ohne sich gegenseitig zu stören.

Welch mächtiges Arbeitsmittel Drag & Drop sein kann, demonstriert Adobe in seiner neuen Produktreihe auf eindrucksvolle Weise: Beispielsweise kann ein Bild im Photoshop unter Beibehaltung sämtlicher Meta-Informationen (Ebenen, Kanäle und so weiter) in ein geöffnetes Illustrator- oder InDesign-Dokument gezogen werden. Auch eine in InDesign platzierte Illustrator-Grafik bleibt voll bearbeitbar und fügt automatisch ihre Ebeneninformationen in die DTP-Software ein.

Ähnlich imposante Ergebnisse lassen sich innerhalb der Microsoft-Produktfamilie aufzeigen. Vor allem zwischen den einzelnen Applikationen der Office-Tools sind hervorragende Drag-&-Drop-Mechanismen implementiert.

Doch hier zeigt sich die Schwäche der proprietären Insellösungen: Mag die Funktion auch die Arbeit innerhalb der eigenen Produktfamilie erleichtern und ein famoses Zusammenspiel ermöglichen, der Schritt dar-

über hinaus ist nicht möglich. Entweder kann die herstellerfremde Anwendung die ankommenden Daten gar nicht oder nur sehr beschränkt interpretieren. Im ersten Fall ist eine Kommunikation auf diesem Wege schlicht unmöglich, im zweiten können nur einfachste Datenstrukturen, wie zum Beispiel 8-Bit-ASCII-Texte, übertragen werden. Doch damit nicht genug: Die Funktionalität muss so betriebssystem- und programmstrukturnah programmiert sein, dass bei einer Anwendung im deutschen Sprachraum sogar mit noch weiteren Komplikationen zu rechnen ist. So kann es durchaus vorkommen – da die meisten Programme in den USA entwickelt werden beziehungsweise die Programmiersprachen auf der englischen Sprache basieren –, dass für solche Transfers die 7-Bit-ASCII-Codierung zur Anwendung kommt. Bei der Übersetzung des Programms in die deutsche Sprache ist es nicht möglich, diese so tief implementierte Vereinbarung zu beeinflussen. Als Resultat kann ein Text statt der erwarteten Umlaute und Akzente nach dem Prozess des Drag & Drop stattdessen kryptische Sonderzeichen zeigen.

6.2.2
Versuchsergebnis

6.2.2.1
Quark XPress 5

Sowohl die Windows- als auch die MacOS-Version unterstützen kein Drag & Drop.

6.2.2.2
InDesign 2

InDesign unterstützt Drag & Drop auf zwei verschiedene Arten. Wählt man den Text in Word ganz aus und zieht ihn auf einen InDesign-Text-Rahmen von Fenster zu Fenster, führt InDesign einen RTF-Import durch. (Näheres zum RTF-Import siehe Kapitel 6.4.)

Die zweite Möglichkeit besteht darin, ein Dateisymbol im Betriebssystem zu nehmen und auf einem Textrahmen fallen zu lassen. Hier führt InDesign sogar einen Word-Import durch, der allerdings etwas fehlerbehafteter war als der RTF-Import. (Näheres zum Word-Import siehe Kapitel 6.3.)

6.3 WORDFORMAT-FILTER

6.3.1
Einleitung

Das Word-Format wird direkt in die DTP-Applikation importiert.

Dieser Import-Schritt stellt die höchsten Anforderungen an den Filter. Wie aus Kapitel 4.3.2.3 über das Word-Dateiformat schon ersichtlich, ist ein Zugriff auf die Strukturen der Word-Datei nicht einfach. Die Versuchsergebnisse geben dies auch wieder.

6.3.2
Versuchsergebnis

6.3.2.1
Quark XPress 5

XPress übernimmt alle Formatvorlagen, die im Word-Dokument definiert sind – egal, ob sie benutzt werden oder nicht. Ein Löschen von Hand darf nicht über die Funktion »Unbenutzte Stilvorlagen löschen« ausgeführt werden, sondern sollte einzeln und sukzessive erfolgen, da im ersten Fall die Stilvorlagenzuordnung teilweise völlig durcheinander kam.

Die Verwendung von zeichenbasierten Stilvorlagen unterscheidet sich stark von der Art und Weise, wie Word sie benutzt. Während Word eine Zeichenvorlage als Zusatzauszeichnung zur entsprechenden Absatzstilvorlage ansieht, geht Quark XPress strikt nach der Stilvorlagendefinition. Ist in Word eine Formatvorlage beispielsweise als »Kursiv« definiert, ist die Schriftgröße abhängig von der zugrunde liegenden Absatzvorlage. In Quark XPress wäre so etwas nicht möglich. Die Definition umfasst hier weit mehr, als nur die Funktion »Kursiv«. Daraus folgt, dass für den Import in XPress für jede Kursivierung eine eigene Stilvorlage angelegt werden muss – und das auch in Word, obwohl es anders möglich wäre.

Die Index-Einträge und Hyperlinks gehen komplett verloren, obwohl Quark XPress 5 beides unterstützt. Ebenso verhält es sich mit den Tabellen. Sie werden in Tabulator-getrennten Text umgewandelt und nicht als Tabelle wiedergegeben.

Die automatische Kapitelnummerierung und automatische Aufzählung werden gar nicht interpretiert. Die Kapitelnummerierung fehlt gänzlich, bei den Aufzählungen fehlt das Aufzählungszeichen. Die Fußnotenziffern und -einträge werden jedoch korrekt inklusive ihrer Formatvorlage interpretiert.

XPress importiert das feste (= nicht zu trennende) Divis (DEC-ASCII 30) als »Einzug hier«-Zeichen, das geschützte Leerzeichen als Viertelgeviert. Das Summenzeichen wurde von der Windows-Version als Fragezeichen umgesetzt, was wieder den Hinweis gibt, dass eine Konvertierung über das System-Encoding erfolgt. In der Macintosh-Version wurde das Summenzeichen korrekt interpretiert.

Ansonsten konnten keine Unterschiede zwischen den beiden Plattformen festgestellt werden.

6.3.2.2 InDesign 2

InDesign gerät beim Import der Formatvorlagen in Schwierigkeiten. Obwohl während des Imports in der Stilvorlagenpalette zu sehen war, dass alle Stilvorlagen durchgeprüft wurden, wurden einige Stilvorlagen automatisch entfernt, obwohl sie im Dokument benutzt wurden. Die Software hat anscheinend Probleme, wenn sich viele Zusatzauszeichnungen im Dokument befinden. Nach Löschen der ersten Zeile, in der viele Zusatzauszeichnungen händisch vorgenommen wurden, geschah der Formatvorlagenimport völlig korrekt. InDesign belässt nur die wirklich verwendeten Stilvorlagen im Dokument. Wünschenswert wäre eine Auswahlmöglichkeit, ob nicht doch auch einige unbenutzte Formatvorlagen übernommen werden sollen.

Einige der Formatvorlagen wurden von ihrer typografischen Formatierung her falsch übernommen. Beispielsweise erschien eine Formatvorlage als Kursive, obwohl dies im Word nicht so definiert war.

Die Windows-Version wirft sogar einige Stilvorlagen komplett durcheinander. Weil hier auch einige Überschriftenhierarchien nicht korrekt erkannt werden, fehlen teilweise auch Kapitelnummerierungen und natürlich deren Stilvorlagen.

Es kann nicht ausgewählt werden, was beim Import mit bereits bestehenden Stilvorlagen in InDesign passieren soll. Die Standard-Einstellung lautet hier, dass die im InDesign definierten verwendet werden. Doch leider funktioniert dies nicht korrekt. Das Programm scheint die Formatdefinitionen aus Word als Abwandlung zur eigentlichen InDesign-Stilvorlage anzusehen. Es erscheint bei allen Absätzen und Zeichen ein »+« hinter der Stilvorlage, was darauf hinweist, dass der ausgewählte Text von der Stilvorlagendefinition abweicht. Nachträgliche Änderungen an Stilvorlagen werden zwar korrekt ausgeführt, doch bleibt ein Rest Verunsicherung. Das »+« zeigt im Normalfall an, dass der ausgewählte Text zwar grundsätzlich mit dieser Stilvorlage ausgezeichnet ist, aber abweichend dazu per Hand formatiert wurde. Damit ist die Zuordnung nicht sauber erfolgt.

Die Definition von zeichenbasierten Stilvorlagen ist der von Word identisch. Es ist im InDesign möglich, eine Stilvorlage anzulegen, die lediglich auf »Kursiv« (elektronische Kursivierung) schaltet. Als Grundlage für die Schriftgröße wird dann die übergeordnete Absatzvorlage hinzugezogen. Dies erleichtert den Texttransfer von Word nach InDesign erheblich.

Das feste Divis setzt InDesign leider nur als normales Divis um.

Alle anderen Word-Funktionen werden jedoch komplett unter Beibehaltung von Index-Einträgen, Hyperlinks, Tabellen und platzierten Bildern übernommen.

6.4 RTF-Filter

6.4.1 Einleitung

Aus Word heraus muss ein RTF-Export erzeugt werden, der in die DTP-Applikation importiert wird.

Wie aus Kapitel 4.3.1.1 ersichtlich, kann das RTF die wichtigsten Funktionalitäten des Word-Formats abbilden. Importiert eine Applikation also das RTF vollständig, bleiben beim Import nahezu alle wichtigen Funktionen erhalten. Die Qualität hängt demnach von der Leistungsfähigkeit des RTF-Filters ab.

6.4.2
Versuchsergebnis

6.4.2.1
Quark XPress 5

Wie in Version 4.11 liefert XPress für das MacOS 9.x keinen RTF-Importfilter aus. Dass ein so wichtiges Format nicht unterstützt wird (oder nur mit zusätzlicher Investition in eine XTension), ist ein großer Nachteil.

Der Windows-Importfilter brachte bei der Testdatei einen Fehler. Da die neuesten Word-Versionen in dieser Arbeit verwendet wurden, ist davon auszugehen, dass der Filter eine veraltete Version von RTF und nicht die neueste unterstützt. Damit hinkt XPress auch hier weit den aktuellen Anforderungen hinterher.

6.4.2.2
InDesign 2

Während der Import auf Windows viele Anforderungen bravourös meistert, schleichen sich in die Mac-Variante einige Fehler ein.

Die Windows-Version importiert sauber alle benutzten Formatvorlagen. Doch gilt zu bemerken, dass die Word-Standard-Formatvorlagen wie »Überschrift 1«, »Normal« und so weiter in englischer Sprache in der DTP-Applikation ankommen. Die Namen der selbst definierten werden korrekt übernommen. Der Schönheitsfehler ist allerdings nicht InDesign, sondern Word anzurechnen. Beim RTF-Export in Word greift das Programm offenbar auf die englische Sprachdefinition zurück und speichert so die Formatvorlagen in das RTF-Dokument. Ein korrekter Export würde hier auch das gewünschte Ergebnis liefern.

Anzumerken gilt, dass auch hier beim Import von selbst definierten Formatvorlagen in ein Dokument mit bereits definierten Stilvorlagen ein »+« hinter der Stilvorlage erscheint. Es ergibt sich also die gleiche Problematik wie beim Wordformat-Filter: Nachträgliche Änderungen an der Stilvorlage werden zwar auf den Text übertragen, doch ist die Zuordnung nicht sauber gelungen.

Alle anderen Testmerkmale wurden korrekt übersetzt: Hyperlinks und die Verschlagwortung mittels Indices bleiben voll erhalten, platzierte Bilder werden importiert, Aufzählungen, Tabellen – alles wird völlig korrekt umgesetzt.

Die Macintosh-Variante verliert beim Import den Gedankenstrich. Die Formatvorlagen werden zwar wie in der Windows-Version übernommen, doch ist ihre Formatierung – wie sie im Word vordefiniert war – völlig verschoben.

6.5 NUR-TEXT-FILTER

6.5.1
Einleitung

Aus Word heraus muss eine Nur-Text-Datei erzeugt werden, die in die DTP-Applikation importiert wird.

Bei dieser Import-Methode kommt es nicht nur auf die Import-Fähigkeit der DTP-Applikation, sondern ebenso auf die Export-Qualität von Word an. Mit welchem Encoding und mit welcher Zeilenendschaltung kann exportiert werden? Wie werden die verschiedenen Word-Funktionalitäten in ein Format übersetzt, dass keinerlei Unterstützung dafür zur Verfügung stellt, wie beispielsweise für Indices, Fußnoten und Hyperlinks? Tabellen werden von Word beispielsweise als absatzgetrennter Text exportiert. Ein Umstand, der die händische Formatierungsarbeit zusätzlich erschwert.

Einzüge versucht Word mittels Leerzeichen am Absatzanfang zu simulieren. Ein Umstand, der lästige Suchen-Ersetzen-Läufe in der DTP-Software zur Folge hat, um die Leerzeichen wieder zu entfernen.

Des Weiteren hängt der Zeichenumfang direkt mit dem verwendeten Encoding zusammen. Speichert man die Datei ANSI-codiert ab (also im Windows-Standard-Encoding), lässt sich das Summenzeichen nicht darstellen, weil es in der Code-Definition nicht vorhanden ist.

Grundsätzlich kann gesagt werden, dass die DTP-Applikationen ein Nur-Text-File problemlos und vollständig importieren können. Doch interessanterweise ergaben sich kleine Interpretationsdifferenzen.

6.5.2
Versuchsergebnis

6.5.2.1
Quark XPress 5

XPress importiert das feste Divis als »Einzug hier«-Zeichen. Da XPress auf das jeweilige Systemencoding eingeschränkt ist, konnte in der Windows-Version kein Summenzeichen (∑) importiert werden, was über das Macintosh- oder Unicode-Encoding kein Problem dargestellt hätte.

6.5.2.2
InDesign 2

Das feste Divis ergab bei der Windows-Version einen sofortigen Abbruch des Imports. Erst das Entfernen des Zeichens brachte den gewünschten Erfolg. Die Macintosh-Version importierte das feste Divis jedoch anstandslos.

InDesign unterstützt eine große Anzahl an Encodings für den Nur-Text-Import. Eine ANSI- oder eine Unicode-codierte Datei aus einem Windows-System kann ohne Weiteres direkt in InDesign platziert werden, was lästige Konvertierungsarbeit erspart.

6.6 XML

Aus Word heraus muss eine XML-Datei erzeugt werden, die in die DTP-Applikation importiert wird.

6.6.1
XML vs. XML

XML kann in unterschiedlichen Stufen zum Einsatz kommen. Soll es in vollem Umfang genutzt werden, verlangt dies ein großes Wissen verschiedenster Bereiche: Allen voran über die Meta-Sprache XML als solche, dann über die Schnittstellenprogrammierung zwischen verschiedenen Systemen und Programmen, über die technische Umsetzung und schlussendlich über die logische Struktur des Inhalts[48]. Da eine Person allein dieses enorme Spektrum nicht abdecken kann, werden die meisten XML-Implementationen von einem Projektteam realisiert.

Neben den fachlichen Kompetenzen sind auch immer psychologische Aspekte entscheidend. Einem Redakteur, Lektor oder Autor ein fertig gestricktes XML-Konzept zu präsentieren, ohne sie an der Entwicklung zu beteiligen, wäre grundverkehrt. Sie alle müssen um die Problematiken und Methoden wissen, um mehr Verständnis für die Vorgaben aufbringen zu können, die ihnen von der Produktion auferlegt werden. Damit ist es wichtig, sie von Anfang an einzubeziehen.

Verschiedene Verlage bringen Autorenrichtlinien heraus, in denen die Grundkonzepte ihrer Produktionswege erläutert werden. (Diese stehen meist auf den Homepages der Verlage zum Download zur Verfügung.) Doch dürfte die Nutzung eines solchen Dokuments recht kärglich ausfallen. Viel sinnvoller wäre es, auch Redakteure, Lektoren und vor allem Autoren grundlegend über XML und den erarbeiteten Workflow in einer Schulung zu informieren. Natürlich muss ein Interesse und ein Wollen beim Autor vorhanden sein. Sonst nutzt die beste Schulung nichts.

Die soeben angesprochene Nutzung von XML deckt sich vollständig mit der in Kapitel 4.3.1.2 beschriebenen Idee, die hinter den Metasprachen steht, Inhalt, Struktur und Formatierung scharf voneinander zu trennen. Natürlich kann XML auch für Produktion benutzt werden, indem die klare Trennung verwaschen wird. Es wird nicht mehr wichtig, dass die Struktur inhaltlich und logisch stringent ist, sondern, dass die Auszeichnung sich dem Produktionsweg maximal anpasst. Tags werden dazu genutzt, Formatierung zu beschreiben, oder codieren Steuerbefehle, die für bestimmte Systeme notwendig oder einfach zu handhaben sind. Damit verlässt man den Grundgedanken der System- und Applikationsunabhängigkeit zugunsten eines bestimmten Workflows.

Für viele Workflows mag das eine ausreichende Markup-Variante darstellen. Der erforderliche Mehraufwand für die korrekte Strukturierung der Daten erhöht die Produktionskosten, bringt in diesen Fällen aber keinen sichtbaren Zusatznutzen – zumal die erfassten Daten nur als Basis genutzt werden, um sie in das proprietäre Format der Software zu konvertieren.

48 Eine Strukturierung des Inhalts kann nur von Menschen vorgenommen werden. Eine Maschine kann die kontextuellen Zusammenhänge nicht über Boolsche Logiken erschließen. Dafür fehlen ihr Mustererkennung, Interpretationsfähigkeit und Fehlertoleranz, die für eine solche Arbeit notwenig sind.

Was vordergründig Vorteile und Erleichterungen bei der Implementierung des Workflows bringt, birgt noch weitere Nachteile: Die Datenbestände müssen bei einem Workflow-, Ausgabemedien- oder Dienstleisterwechsel umgearbeitet werden. Die vorher gescheuten Kosten fallen unweigerlich an (wenn nicht sogar noch höhere). Die Möglichkeit, aus alten Datenbeständen mit den neueren Versionen von DTP-Software automatisch XML zu exportieren, erzeugt wiederum nur »minderwertiges« XML – eben jenes XML, das lediglich Steuerbefehle codiert und nicht Struktur, Inhalt und Formatierung sauber voneinander trennt. Es bleibt damit applikations- und medienabhängig, denn aus einer formatierungsorientierten Applikation lässt sich auf automatischem Wege kein strukturiertes XML extrahieren.

Wie weit die Philosophie hinter den Auszeichnungssprachen in einem Workflow beachtet wird, ist eine Entscheidung, die gut bedacht werden muss. Es ist keine rein produktionstechnische, sondern schon eine (unternehmens-)strategische Entscheidung.

6.6.2 DTP-Programme und XML

Die beiden Programme behandeln XML unterschiedlich. Doch es gibt auch Gemeinsamkeiten: Sie können keine Transformationen erzeugen.

Ein Beispiel. In einer XML-Instanz ist folgende Zeile enthalten: `<tipp>Die Schraube <artikel idnr="1001-25"/> immer vorsichtig lösen!</tipp>`

Die formatierte Darstellung soll wie folgt aussehen: »*Tipp:* Die Schraube M25 immer vorsichtig lösen!«

Um diese Aufgabe zu bewältigen, müsste die DTP-Applikation neben der Formatierung noch die folgenden zwei Dinge leisten: Vor alle `<tipp>`-getaggten Textpassagen ein »Tipp: « einfügen und den Link auf einen Artikel auswerten, den Namen des Artikels auslesen (aus einer Datenbank oder anderen XML-Quelle) und an dieser Stelle ausgeben. Solche Transformierungsschritte sind derzeit mit beiden Programmen nicht zu realisieren. Damit die oben geforderte Ausgabe erfolgt, muss zuvor eine Transformation per Suchen/Ersetzen durchgeführt werden, die sogar ein neues Tag einführen muss: `<tipp><tkrs>Tipp:</tkrs> Die Schraube M25 immer vorsichtig lösen!</tipp>`

Ein weiteres Beispiel sind Zeilenendschaltungen. Idealerweise sind im XML-Strom die Zeilenendschaltungen nur der Übersichtlichkeit wegen getastet und werden vom verarbeitenden Programm nicht berücksichtigt. Wie viele Leerzeilen beispielsweise nach einem Absatz folgen, sollte in jedem Fall über das XSLT- oder DSSSL-Skript realisiert werden und nicht in der XML-Instanz. Da die DTP-Programme keine Transformierung beherrschen, müssen sämtliche Zeilenendschaltungen in die XML-Instanz gebracht werden.

6.6.2.1 Quark XPress

Quark XPress arbeitet mit so genannten Platzhaltern, die für einen bestimmten Elementtyp stehen und damit letztendlich den Inhalt eines bestimmten Tags vertreten. Diese Platzhalter werden formatiert, der Inhalt fließt beim Import automatisch ein. Der XPress-Anwender kann den XML-Inhalt nur über die Platzhalter formatieren.

Quark XPress bietet die Möglichkeit, Platzhalter, denen XML-Inhalt zugewiesen ist, in Text zu verwandeln. Damit verliert die XML-Datei ihre XML-Eigenschaften und wird zu normalem XPress-Text. Dieser erlaubt nun wieder, einzelne Zeichen beliebig zu manipulieren. Bei der Verwendung von Platzhaltern kann nur elementweise formatiert werden.

Eine Instanz wird zunächst komplett geladen und in der Platzhalter-Palette angezeigt. Sie muss keinem Rahmen direkt zugeordnet werden. Welcher Text in welchen Rahmen geladen werden soll, kann der Anwender frei bestimmen, in dem er die entsprechenden Platzhalter in den entsprechenden Rahmen zieht.

Quark XPress arbeitet nur mit validen XML-Instanzen, das heißt, es muss eine DTD vorhanden sein und die Dokument-Instanz der DTD entsprechen. Demnach

validiert XPress die Instanz vor dem Import. So lobenswert dieser Ansatz ist, es ergeben sich damit weitere Fehlerquellen, die aus der Programmierung von Quark XPress herrühren. Einige XML-Konstrukte in DTDs sind unzulässig und werden den Import mit einem Fehler beenden. Ein Beispiel ist die Verwendung von Parameter-Entities.

Sie werden verwendet, um eine DTD übersichtlicher zu gestalten. Sie sind Abkürzungen für konstante Ausdrücke, stehen also als Kurzschreibweise für einen anderen Inhalt. Bei der Verarbeitung der XML-Datei werden alle Parameter Entities vom Parser durch den regulären Ausdruck ausgetauscht.

Eine DTD enthält beispielsweise folgende Definition einer Parameter Entity:

```
<!ENTITY % blocks "list | section |
table">
```

Ab dieser Definition kann in der DTD folgende Schreibweise verwendet werden: `%blocks;` Sie steht für den Ausdruck: `list | section | table`.

Die nachfolgende Element-Definition:

```
<!ELEMENT par  (%blocks;* | p)>
```

wird bei der Ausführung und Bearbeitung ersetzt durch:

```
<!ELEMENT par  ((list | section |
table)* | p)>
```

Je komplexer eine DTD wird, desto sinnvoller ist die Anwendung solcher Parameter Entities.

Auch die Verwendung von XML-konformen Kommentaren in der DTD brachte Import-Schwierigkeiten.

Der Import hält demnach eine Fülle an Schwierigkeiten und Stolperfallen bereit, die ein tiefes Wissen um XML erfordern.

6.6.2.2
Adobe InDesign

InDesign behandelt XML-Dokumente anders. Zunächst wird – wie in XPress – die komplette Instanz geladen, ohne dass sie einem bestimmten Rahmen zugeordnet werden muss. Es gibt einen so genannten structure view, der die komplette Instanz mit allen Elementen und Attributen hierarchisch gegliedert anzeigt. Auf Wunsch lässt sich in dieser Ansicht auch der Element-Inhalt anzeigen. Der Anwender kann nun selbst entscheiden, welcher Teil der Instanz welchem Rahmen zugeordnet werden soll. Sie kann auf diese Weise recht komfortabel in komplexe Layouts übertragen werden.

Um die Instanz zu formatieren stehen zwei Möglichkeiten zur Verfügung. Entweder kann der Anwender manuell direkt im platzierten Inhalt formatieren oder Stilvorlagen den Tags zuweisen. Bei letzterem besteht noch die Wahl zwischen Formaten den Tags zuzuweisen oder umgekehrt. Sehr komfortabel ist der Befehl, eine Zuordnung nach Namen vorzunehmen. Sind also alle Stilvorlagen so benannt, wie ihre korrespondierenden Tags, ist die Zuordnung darüber besonders komfortabel.

Dem Anwender stehen so genannte templates zur Verfügung, mit deren Hilfe ein Layout komplett vordefiniert werden kann. Sobald der XML-codierte Text dann zur Verfügung steht, fließt er automatisch in die vordefinierten Bereiche ein.

InDesign arbeitet nur mit well-formed XML-Instanzen, das heißt, eine DTD muss nicht vorhanden sein. Eine Validierung, also Gültigkeitsprüfung, kann zuvor über Skripte oder andere Programme erfolgen.

6.6.3
UpCast

Der sauberste Weg aus einer »richtigen« XML-Datei im Sinne des Kapitels 4.3.1.2 in eine DTP-Applikation, sollte wegen der höheren Leistungsfähigkeit eine Transformation der XML-Instanzen mittels eines XSLT- oder DSSSL-Skripts in XPress-Marken beziehungsweise Tagged Text erfolgen. Zwar könnte dies ein geschickter Programmierer mit einem VBA-Word-Makro ebenfalls erwirken, doch stehen in XSL und DSSSL die benötigten Algorithmen schon zur Verfügung, wohingegen sie bei einem VBA-Makro erst nachprogrammiert werden müssten. Dieser Weg lohnt sich aber nur bei wiederkehrenden XML-Strukturen oder der Publikation auf mehrere Medien. Für die Erstellung einer einmaligen Print-Publikation wären die Entwicklungskosten horrend.

Da diese Diplomarbeit den Weg aus einem Word-Dokument in die DTP-Applikation untersucht, muss aus Word zunächst eine XML-Instanz erzeugt werden.

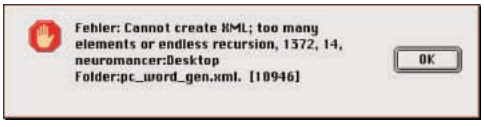
UpCast soll hier exemplarisch für eine Reihe von Programmen stehen, die aus Word-Dateien XML exportieren. Weitere wichtige Vertreter sind Mediatext oder SGML-Author.

Die ersten zwei Seiten der Testdatei sind im Anhang unter Kapitel 9.5.2 abgedruckt.

6.6.4 Versuchsergebnis

6.6.4.1 Quark XPress 5

Selbst nach Beseitigung aller Parameter Entities und dem Entfernen aller XML-Kommentare war XPress nicht in der Lage, die DTD zu verarbeiten. Es erschien die Fehlermeldung, dass zu viele Elemente definiert seien oder das Dokument Endlosschleifen enthalte:



Damit ist es – selbst nach kurzer manueller Überarbeitung – nicht möglich gewesen, die von UpCast erzeugte DTD und XML-Instanz zu übernehmen.

Um herauszufinden, wo der Import Schwierigkeiten bereitet, muss der Anwender tief in die Struktur der DTD einsteigen und manuell eingreifen.

6.6.4.2 InDesign 2

InDesign importierte den XML-Strom ohne Schwierigkeiten und stellte die Struktur völlig korrekt im structures view dar. Das Übertragen in einen Textrahmen gelang ohne Probleme. Die Zuordnung von Formatierung muss – falls nicht über ein template gearbeitet wurde – von Hand erfolgen.

Indexeinträge und Hyperlinks konnten auf diesem Weg nicht direkt als solche importiert werden, obwohl sie von UpCast korrekt in die XML-Syntax übertragen wurden. Eine solche Zuordnung muss der Anwender im Nachhinein selbst erledigen.

Auch die Tabellenbehandlung ließe sich verbessern: Obwohl das Tabellenmodell von InDesign sehr XML-konform definiert ist, gibt es keine Möglichkeit, aus einem XML-Strom automatisch Tags zu einer Tabelle zusammenfassen zu lassen. Alle Tabellen müssen im Nachhinein vom Anwender per Hand aufbereitet werden.

Wurden die Umlaute korrekt in ihre escape notations übertragen, blieben Summenzeichen und Euro unübersetzt. Auch wenn die Datei im Unicode-Encoding abgespeichert war und dies damit eigentlich nicht nötig ist, ist eine solche Codierung empfehlenswert, um die Produktionssicherheit zusätzlich zu erhöhen. InDesign jedoch setzte alles korrekt um.

Sehr komfortabel werden alle Auszeichnungen per Hand behandelt. Sie erhielten bei der Konvertierung von UpCast ein eigenes Tag mit Attribut: `<inline kind="italic">...</inline>` Damit kann jeder Auszeichnung eine eigene Stilvorlage zugeordnet werden.

InDesign erlaubt – im Gegensatz zu Quark XPress – das direkte Anwählen und Formatieren von einzelnen Zeichen, ohne den XML-Strom in reinen Text mit Stilvorlagen konvertieren zu müssen. Damit hat Adobe die strikte Trennung von Struktur und Formatierung umgesetzt, was sehr zu begrüßen ist.

Weit schwieriger lassen sich die Fußnoten aus dem XML-Strom extrahieren. Sie sind – was völlig mit dem Gedanken von XML übereinstimmt – direkt hinter dem Wort codiert, das den Fußnotenverweis enthält und damit über die ganze Instanz verteilt. Die Fußnotenverweise sind nicht mehr durchnummeriert wie in Word, sondern haben eine eigene Referenznummer erhalten. Das Codierungssystem setzt ein Programm voraus, das Fußnoten eigenständig verwalten und XML-Inhalte transformieren kann. Fußnotennummern müssen von der Applikation neu vergeben und automatisch verwaltet werden. All das kann InDesign nicht leisten, so dass die Aufbereitung der Fußnoten sehr schwierig und zeitaufwändig wird.

Völlig unsinnigerweise übernimmt InDesign getreu die Syntaxeinrückung der XML-Datei mit allen Leerzeichen und Umbrüchen. Wie schon im Kapitel über SGML/XML dargelegt, müsste die genaue Ausstattung des Inhalts mit Umbrüchen durch eine Transformation geschehen. Dass InDesign aber Text zwischen zwei Tags übernimmt ist trotzdem nicht einzusehen. Dieser sollte konsequent ignoriert werden. Am besten eignet sich also zum Import ein durchgängiger XML-Strom, der keinerlei zusätzlichen Zeichen (keine Umbrüche oder Leerzeichen zur Syntaxeinrückung) zwischen Tags enthält.

Zu bemängeln ist außerdem eine deutliche, fast schon unerträgliche Verlangsamung des Programms bei der Handhabung der 488 KB großen XML-Datei.

6.7 Steuercodes für DTP-Programme

Aus dem Word-File wird ein Strom von Nur-Text-codiertem Text erzeugt, der spezielle Steuerzeichen enthält, um ihn in InDesign oder Quark XPress importieren zu können.

Die Idee, Texte mittels Steuerbefehlen zu codieren, ist nicht neu, sondern stellt den Beginn der digitalen Satzverarbeitung dar. Anfangs gab es noch kein WYSIWYG (what you see is what you get), sondern die bloße Darstellung des Textes, der mittels der Steuerbefehle »gestaltet« wurde. Später kamen so genannte Makros[49] hinzu: Eine wiederkehrende Folge von Texten und/oder Befehlen wurde im System hinterlegt und bei Bedarf mit einem Kurzaufruf aktiviert.

Quark XPress und Adobe InDesign lassen sich ebenfalls mit solchen Textcodierungen steuern. Der Weg ist bei beiden Programmen der gleiche: Eine ASCII-Datei wird mit den Steuerbefehlen versehen und über einen speziellen Filter in die DTP-Software importiert. Das Programm setzt dann automatisch die enthaltenen Befehle in die entsprechende typografische Formatierung um.

Natürlich sind es zwei verschiedene Codierungssprachen, die untereinander nicht kompatibel sind. Um einen XPress-Marken-codierten Text in InDesign zu importieren, muss dieser erst konvertiert werden und umgekehrt.

Beide Sprachen unterscheiden sich auch mnemotechnisch sehr stark. Sind also beide Systeme im Einsatz, eventuell noch ein Satzsystem wie PageOne oder Berthold, wird die Texterfassung und Arbeitsvorbereitung schnell unübersichtlich und kompliziert. Manuskripte müssen mit mehreren Auszeichnungssprachen codiert, bei der Erfassung mehrere Auszeichnungssprachen erfasst werden. Fehler sind vorprogrammiert. Ist ein Text einmal in einer Codierung erfasst worden, besteht keine Möglichkeit mehr, das System kurzfristig, beispielsweise aus Kapazitätsgründen, zu wechseln.

Abhilfe schafft hier nur eine Vereinheitlichung, die Einführung einer allgemeinen Markup-Language, die mit einer Konvertierung flexibel in die verschiedenen Codierungssprachen übersetzt werden kann.

Dies ist genau der Punkt, an dem Markup-Languages – wie in Kapitel 6.6.1 beschrieben – minimalistisch und nicht ihrer eigentlichen in Kapitel 4.3.1.2 dargestellten Philosophie entsprechend eingesetzt werden: Die Tags codieren proprietäre Steuerbefehle und unterwerfen sich somit maximal dem Workflow.

6.7.1 XPress-Marken

Die meisten XPress-Marken beginnen mit einem TAGO (tag open), dann folgt der Formatierungsbefehl und enden mit einem TAGC (tag close). Je nach Befehl stehen dem TAGC noch Attribute voran. Es gibt allerdings keine einheitliche Syntax. Einige stehen ohne zusätzliche Zeichen direkt hinter dem Befehl, andere müssen in Zollzeichen eingeschlossen werden, wieder andere stehen Kommata-getrennt in runden Klammern. Endet einer der Befehle mit einem Dollar-Zeichen ($), wird diese Formatierung auf den in der aktuellen Stilvorlage vorgegebenen Wert zurückgesetzt, also aufgehoben (siehe im Beispiel für Zeichenvorlagen).

Absatzvorlagen müssen an den Anfang des Absatzes gesetzt werden. Sie folgen der BNF-notierten Syntax[50]:

```
'@' Stilvorlagenname ':'
```

49 Der Name war nicht normiert, sondern hieß in jedem System anders: Befehlskette, IWT (immer wiederkehrender Text), Data, Kombination, Typonummer, Makroinformation, Multicode, Softkey, Wenn-immer-Taste, User Defined Keys und viele mehr.

50 Zur BNF siehe Kapitel 9.5

Beispielsweise: `@Grundtext:Nennt mich Ismael. [...]`

Zeichenvorlagen werden direkt an den gewünschten Textstellen eingebracht. Sie folgen der Syntax: `'<' '@' Stilvorlagenname '>'`

Beispielsweise: `Um es kurz zu machen: Der Wal ist <@Grundschrift kursiv>ein Sprühstrahlen ausspritzender Fisch mit waagerechtem Schwanz<@$p>. Da habt ihr ihn.`

Die Datei kann mit einem Zeichensatz-Tag begonnen werden, um das Encoding festzulegen. Dabei stehen die folgenden drei Encodings zur Verfügung (die Zahlen in Klammern entsprechen der zu setzenden Zahl): Mac- (`0`), Windows- (`1`) und ISO-Latin-1-Zeichensatz (`2`). Die Syntax lautet: `'<' 'e' Zahl '>'`

Beispielsweise:

```
<e0>
@Überschrift1:1
Ausblick
@Grundschrift:Nennt mich Ismael. [...]
```

Eine vollständige Referenz der XPress-Tags ist im XPress Reference Book der Dokumentation enthalten.

6.7.2 Adobes Tagged Text

Ausschließlich alle InDesign-Tags beginnen mit einem TAGO, gefolgt vom Befehlsnamen mit abschließendem Doppelpunkt. Dahinter befinden sich die Attributswerte, die ohne Anführungszeichen zu setzen sind, und das TAGC: `<cFont:Minion Bold OsF>`

Wird ein Absatzformat zugewiesen, muss der Befehl am Beginn eines Absatzes stehen: `<ParaStyle:Name>`. Er gilt so lange, bis er von einem anderen ParaStyle abgewechselt wird und folgt der Syntax: `'<ParaStyle:' Formatname '>'`

Beispielsweise: `<ParaStyle:Grundschrift>Nennt mich Ismael. [...]`

Zeichenformate hingegen werden direkt an den gewünschten Stellen in den Text eingebracht. Sie beginnen mit `{'<CharStyle:' Formatname '>'}`[51] und enden mit `{'<CharStyle:>'}` oder mit Beginn eines neuen Zeichenformats. Beispielsweise: `Um es kurz zu machen: Der Wal ist <CharStyle:Grundschrift kursiv>ein Sprühstrahlen ausspritzender Fisch mit waagerechtem Schwanz<CharStyle:>. Da habt ihr ihn.`

Eine Formatierung ohne Stilvorlagen bleibt so lange wirksam, bis sie mit einem anderen Wert überschrieben wird beziehungsweise mit Hilfe eines Codes außer Kraft gesetzt wird. Als Beispiel wieder das Unterstreichen eines Wortes:

```
[...] gibt es <cUnderline:1>nichts
<cUnderline:>, was ich [...]
```

Wie in XPress ist der Export (diesmal als »Tagged-Text«) die einfachste Art, zu den gewünschten Auszeichnungen zu kommen. Aus der Datei, die InDesign erstellt, können dann alle benötigten Steuerzeichen in ihrer genauen Anwendung herausgelesen werden.

Damit eine Datei von InDesign als Tagged Text erkannt wird, muss sie mit einen Dateianfangs-Tag starten. Es enthält das Encoding und die Plattform. Hier die Syntax:

```
Dateianfang ::= '<' Encoding '-'
                Plattform '>'
Encoding    ::= [ASCII | ANSI |
                UNICODE | SJIS[52]]
Plattform   ::= [MAC | WIN]
```

Die Formatvorlagen müssen in der Tagged-Text-Datei korrekt im Kopf definiert werden, damit sie von InDesign erkannt und angelegt werden. Dies geschieht über die Befehle `{'<DefineParaStyle:' Formatname '>'}` und `{'<DefineCharStyle:' Formatname '>'}`, so dass eine vollständige Tagged-Text-Datei beispielsweise wie folgt aussieht:

```
<ASCII-MAC>
<DefineParaStyle:grundschrift>
<DefineCharStyle:kursiv>
<ParaStyle:grundschrift>Um es kurz
zu machen: Der Wal ist <CharStyle:
```

51 Bei Verwendung der BNF im Fließtext, wird diese der Übersichtlichkeit wegen in Akkoladen (`{}`) eingeklammert.

52 SJIS ist ein japanisches Encoding.

```
kursiv>ein Sprühstrahlen ausspritzen-
der Fisch mit waagerechtem
Schwanz<CharStyle:>. Da habt ihr ihn.
```

Eine vollständige Befehlsreferenz für Tagged Text ist auf der Installations-CD von InDesign als PDF enthalten.

6.7.3
Vom Word-Format zu den Steuercodes mit VBA

Die Zuordnung der Word-Felder und -Formatvorlagen zu den entsprechenden Steuerbefehlen für XPress-Marken oder InDesign-Tags kann über Word-Makros geschehen. Die komfortablere Möglichkeit wäre ein Word-Exportfilter für XPress-Marken oder Tagged Text, was bisher allerdings noch nicht realisiert wurde.

Makros in Word sind kleine Sub-Programme, die mit einem einzigen Aufruf in Gang gesetzt werden können. Sie enthalten frei programmierbare Befehlsfolgen, die Word automatisch steuern und damit Texte beliebig bearbeiten können. Der gesamte Funktionsumfang von Word steht dafür zur Verfügung. Die verwendete Programmiersprache heißt VBA, Visual Basic for Applications, und ist sehr leistungsfähig. Schranken ergeben sich also nur durch Kenntnisse und Geschick des Entwicklers.

Bei guter Programmierkenntnis kann ein solches Makro komplett im eigenen Haus erstellt und verbessert werden. Eigenentwicklungen verursachen zwar (Entwicklungs-)Kosten, die nicht direkt zur Wertschöpfung des Unternehmens beitragen, reduzieren aber langfristig Konvertierungskosten, wenn sie optimal auf den Workflow im Haus abgestimmt und sauber programmiert wurden. Sie stellen ein großes Flexibilitätspotential dar, weil die Anpassung direkt und für das Unternehmen in maximaler Erfüllung der Erfordernisse geschehen kann. Die Integration spezieller Algorithmen, die typografische, textliche oder systemtechnische Erfordernisse automatisch erledigen, ist leicht selbst zu programmieren. Beispiele hierfür sind das korrekte Spationieren von Zahlen, komplexe Suchen/Ersetzen-Routinen, das Berechnen von Formeln oder die Ausgabe von bestimmten Variablen.

Ein eingerichteter Workflow mit eigenentwickelten Konvertern kann über Jahre hinweg eingesetzt werden und die Produktion erheblich erleichtern.

Für die Entwicklung von VBA-Makros steht in Word eine eigene Entwicklungsumgebung zur Verfügung: der Visual-Basic-Editor. Makros können programmiert oder aufgezeichnet werden, wobei der Benutzer die gewünschten Verarbeitungsschritte einfach ausführt, während das Programm im Hintergrund den zugehörigen Code generiert. Die Aufzeichnung hilft bei einigen Unklarheiten der Steuerung schnell weiter. Doch für eine maximale Ausnutzung des Funktionsumfangs und guten Code sollte das Programmieren per Hand bevorzugt werden.

Die Sprachgrundlage ist ein Basic-Abkömmling, der objektorientiert weiter entwickelt wurde und einen Befehlsumfang speziell zur Steuerung der Office-Anwendungen bereit stellt. Für eine vollständige Befehlsreferenz, sei auf diverse Microsoft-Publikationen und die Online-Hilfe verwiesen.

Im Internet finden sich zahlreiche Beispielmakros, Newsgroups, Mailing-Listen und Communities, die in schwierigen Fällen schnell, kostenlos und effizient beraten.

6.7.4
tagNow-Beispielmakro

Das Beispielmakro zeigt einige Basisfunktionen auf und verdeutlicht die potentiellen Möglichkeiten und den Komfort, den eine solche Realisierung mit sich bringt. Viele der Funktionen könnten natürlich noch leistungsfähiger und nutzerfreundlicher programmiert werden, einige fehlen ganz, obwohl sie realisierbar sind. Der gut kommentierte Quelltext des Makros findet sich im Anhang unter Kapitel 9.6. Aufbau und Funktionalität sollen hier aber näher erläutert werden.

Um einen umfassenderen Eindruck der Möglichkeiten der Markoprogrammierung zu erhalten, sei auf den mm²-Editor von Mazzetti & Mazzetti[53] verwiesen. Der Editor ist nichts anderes als eine Ansammlung von Makros, die in mehreren Word-Dokumentvorlagen gespeichert sind.

53 Online ist eine kostenlose Light-Version downloadbar: http://www.mm2-editor.com, Stand: 21.10.2002.

Um das Makro ausführen zu können, müssen einige Grundvoraussetzungen geschaffen werden. Das Makro sollte in der Vorlage »Normal.dot« gespeichert werden, um global für alle geöffneten Dokumente zur Verfügung zu stehen. Es wird eine Word-Datei benötigt, die das auszuzeichnende Dokument enthält, und eine Nur-Text-Datei, die die Auszeichnungsregeln beschreibt, also die Tagdefinitionen enthält.

Für das Beispielmakro muss ein korrekt angelegtes Word-Dokument vorliegen, da keine Abweichungen oder Fehlertoleranzen einprogrammiert wurden. Ein solches Dokument muss vollständig mit Formatvorlagen ausgezeichnet sein. Alle manuellen Formatierungen bleiben unberücksichtigt und gehen verloren.

Die Nur-Text-Datei enthält die Zuordnung von Formatvorlagenname zu Start- und Ende-Tag und die Definition, ob es sich um eine zeichen- oder absatzorientierte Formatvorlage handelt. Die Syntax einer solchen Definition ist wie folgt:

```
NameFormatvorlage <tab> StartTag <tab>
EndeTag <tab> FVArt <return>
```

Während `FVArt` nur die Werte »0« für zeichen- und »1« für absatzorientierte Formatvorlagen einnehmen darf, sind bei allen anderen Merkmalen jedes Zeichen außer Tabulatoren und Returns erlaubt. Der Tabulator wird als Trennzeichen (Delimiter) der einzelnen Merkmale, das Return als Trennzeichen von Zeilen benutzt. Die Datei wird sequentiell eingelesen und verarbeitet.

Im Folgenden wird der Ablauf des Makros näher erläutert:

Der erste Teil erzeugt einen Öffnen-Dialog, mit dem die Auszeichnungsdatei ausgewählt werden kann. Das Makro öffnet sie dann sequentiell, zählt die angelegten Zeilen und dimensioniert damit ein Variablen-Array, das mit den Auszeichnungsinformationen gefüllt wird. Es folgt ein einfaches Beispiel der Handhabung von Tabellen, Aufzählungen und Nummerierungen. Die Anweisungen konvertieren alle im Dokument vorkommenden Tabellen in Tabulator-getrennten Text, alle Aufzählungen und Nummerierungen in Text. Hier sind weit komplexere Verarbeitungsalgorithmen vorstellbar.

Im Nachfolgenden ist das eigentliche Auszeichnen des Word-Dokuments definiert. Zunächst geschieht die Auszeichnung der zeichen-, dann der absatzorientierten Formatvorlagen. Der Algorithmus funktioniert wie folgt: Aus dem Array wird die erste Formatvorlage ausgelesen. Mittels einer Bedingungsprüfung (if) wird festgestellt, ob es sich um eine zeichen- oder eine absatzorientierte Formatvorlage handelt (conArt ist im ersteren Fall 0, im letzteren 1). Anschließend werden die Parameter (Name der Formatvorlage, Start- und Ende-Tag) an die entsprechende Funktion (tagZeichen() oder tagAbsatz()) übergeben.

Die Funktion springt an den Beginn des Dokuments und startet einen Suchen-Lauf: Nacheinander werden zusammenhängende Zeichenfolgen (mit einem oder mehr Zeichen) angesprungen, die mit der gleichen Formatvorlage ausgezeichnet sind. Ist eine Textstelle gefunden, springt der Cursor davor, fügt das Anfangs-Tag ein, markiert wieder die Textstelle, schaltet die Formatvorlage auf Standard um, springt hinter die Auswahl und fügt dort das Ende-Tag ein. Dies wiederholt sich so lange, bis keine Stellen mit dieser Formatvorlage mehr gefunden wurden.

Der Algorithmus lässt Absatzzeichen, Chr$(13), außen vor. In einigen Tests hat sich gezeigt, dass Word bei ausgezeichneten Absatzzeichen in eine Endlosschleife gerät und immer wieder ein und dasselbe Absatzzeichen auszeichnet.

Beim Verarbeiten von absatzorientierten Formatvorlagen muss noch das Dokumentende berücksichtigt werden, damit auch hier keine Endlosschleife auftritt.

Als Abschluss wird der gesamte Text markiert und sämtliche Absatzeinzüge auf Null gesetzt. Dies ist nötig, da die Konvertierung ins Nur-Text-Format Absatzeinzüge mit Leerzeichen simuliert und damit absatzorientierte Stilvorlagenanweisungen nicht mehr ganz am Anfang der Zeile stehen. Sie stehen nach den Leerzeichen und werden damit nicht mehr interpretiert, sondern als Text in der DTP-Applikation dargestellt.

Nun kann der Benutzer die Datei als Nur-Text abspeichern und in Quark XPress oder InDesign importieren.

Bei InDesign muss vor dem Speichern noch der »Dateianfang«-Eintrag (Plattform und Encoding, beispielsweise <MAC-ASCII>) erfolgen, damit die Datei korrekt als Tagged-Text erkannt und verarbeitet wird.

In einer ausführlicheren Version des Makros kann dies über eine Benutzerauswahl, ob das Dokument für XPress oder InDesign ausgezeichnet werden soll, automatisch eingefügt werden. Des Weiteren können viele Hilfen programmiert werden. Zum Beispiel das automatische Generieren und Bearbeiten der Auszeichnungsdatei, damit der Benutzer die genaue Syntax nicht mehr kennen muss. Außerdem können noch typografische Funktionalitäten erdacht werden, wie Spationierung von Zahlen, automatisches Tauschen von doppelten Leerzeichen und mehr.

6.7.5 Versuchsergebnis

Die VBA-Sprache ermöglicht prinzipiell den vollen Zugriff auf die Struktur eines Word-Dokuments. An alle Einzelelemente heranzukommen und sie auszuzeichnen ist somit kein Problem. Die entstandenen Einschränkungen ergeben sich aus der einfachen Tatsache, dass sie im Beispielmakro nicht implementiert wurden.

Bei allen nicht implementierten Funktionen wirkt also der Word-Export-Filter für das Nur-Text-Format mit all seinen Limitationen auf die Konvertierungsqualität.

Die ersten zwei Seiten der Testdatei sind Steuerzeichen-codiert im Anhang unter Kapitel 9.5.3 und 9.5.4 abgedruckt.

6.7.5.1 Quark XPress 5

Alle verwendeten Formatvorlagen werden völlig korrekt und sauber in die Applikation übernommen. Obwohl eine entsprechende Makroprogrammierung Indexeinträge und Hyperlinks finden und auszeichnen könnte, ist es nicht möglich, über XPress-Marken ohne zusätzliche XTensions diese Funktionalitäten zu importieren. Der Benutzer muss die Auszeichnungen nachpflegen.

Quark XPress behandelt Tabellen – im Gegensatz zu Adobe InDesign – als Sonderrahmen und integriert diese nicht in den Fließtext. Die Funktion »Text in Tabelle« nimmt Text aus einem Textrahmen heraus, erzeugt einen Tabellenrahmen, der unabhängig vom ursprünglichen Textrahmen ist, und fügt den ausgewählten Text dort ein. Demzufolge ist es nicht möglich ohne entsprechende XTension, Tabellen automatisch per XPress-Marken übernehmen zu wollen. Diese Art der Tabellenbehandlung ist umso unverständlicher, da sie den Möglichkeiten der Metasprachen SGML/XML völlig widerspricht.

Dass sämtliche Auszeichnungen, die per Hand vorgenommen wurden, nicht importiert wurden, lässt sich durch eine entsprechende Programmierung im Makro leicht korrigieren. Ebenso kann der Import von Sonderzeichen so optimiert werden, dass alle korrekt übertragen werden. Sämtliche Einschränkungen in diesem Bereich rühren aus dem Nur-Text-Export von Word.

Der Import der Fußnoten kann auch noch stark verbessert werden. Eine Übernahme der dortigen Stilvorlagen stellt für VBA kein Problem dar.

6.7.5.2 InDesign 2

Auch für InDesign lässt sich das Makro beliebig verbessern. Mit geschickter Programmierung übertrifft InDesign Quark XPress dann um Längen: Nicht nur der Import der Handauszeichnungen, Fußnoten und Sonderzeichen lässt sich so optimieren, dass alles korrekt umgesetzt wird, über die Steuercodes sind Hyperlinks, Indices und Tabellen in vollem Umfang definierbar. InDesign behandelt Tabellenzellen – wie beispielsweise Word und die Metasprachen SGML/XML – als Spezialfall eines Absatzes, weshalb die Integration völlig problemlos möglich ist.

Wird eine Tagged-Text-Datei in ein fertiges Layout importiert, steht ein Auswahldialog zur Verfügung, ob die in der Importdatei oder in der InDesign-Datei definierten Stilvorlagen bei Namenskonflikten benutzt werden sollen. Die Einstellung ist nur global möglich und nicht – wie in Quark XPress – jeweils pro Stilvorlage.

Alle Stilvorlagen werden korrekt zugewiesen, es erscheint kein »+« in der Stilvorlagenpalette. Die Zuordnung ging sauber vonstatten. Damit ist sichergestellt, dass nachträgliche Änderungen an den Stilvorlagen auf den ausgezeichneten Text korrekt übertragen werden.

Kapitel 7

Der optimale Weg

7.0 EINLEITUNG

Es folgt erneut eine distanzierte Betrachtung der in Kapitel 6 untersuchten Wege von Word in DTP um daraus ein Ergebnis zu entwickeln.

7.1 COPY & PASTE UND DRAG & DROP

Diese eleganten Möglichkeiten des Betriebssystems, Daten zwischen zwei Applikationen auszutauschen, werden von InDesign vorzüglich unterstützt. Die Anwendung führt zumeist den sehr leistungsfähigen RTF-Import durch. Unter MacOS 9.2.1 verhindert das veraltete Betriebssystem die umfangreichen Möglichkeiten. Der Umstieg auf MacOS X sollte hier den gleichen Komfort wie unter Windows 2000 bringen.

XPress nutzt lediglich Copy & Paste und beschränkt sich hierbei komplett auf den Transfer von systemcodiertem, unformatiertem Text, wodurch der gesamte Text per Hand nachformatiert werden muss.

Fügt man systemcodierten, unformatierten Text per Copy & Paste an einer Stelle im Dokument ein, die bereits vorformatiert ist, so wird der Text automatisch mit der vordefinierten Formatierung eingefügt. Will man dieses Feature in InDesign nutzen, muss ein Zwischenschritt eingefügt werden: Der Text wird aus Word kopiert, in einen Texteditor (beispielsweise BBEdit auf dem Macintosh oder Notepad unter Windows) eingefügt, wieder kopiert und in InDesign platziert. Auf diesem Wege gehen alle Formatierungen aus Word verloren. Übrig bleibt die reine Textinformation. Zu beachten gilt lediglich, dass der Texteditor Unicode-Unterstützung bietet, so dass auch alle Zeichen korrekt in InDesign übernommen werden.

7.2 WORD-, RTF- ODER NUR-TEXT-IMPORT

Der Import über RTF oder das Word-Format ist ein guter Weg, wenn noch kein Layout gemacht wurde, da alle Stilvorlagen angelegt werden und der Setzer sie einfach umformatieren kann. Allerdings ist zu beachten, dass sämtliche Formatierungen, die per Hand im Word-Dokument vorgenommen wurden, erhalten bleiben und in die DTP-Applikation übertragen werden.

Ist schon ein Layout mit definierten Stilvorlagen vorhanden, eignen sich RTF- und Word-Import nicht so sehr, da im Word angelegte Formatierungen mit importiert werden. Während in XPress sauber gewählt werden kann, was mit bereits angelegten Stilvorlagen beim Import von gleichnamigen passieren soll, bleibt im InDesign immer eine letzte Unsicherheit, obwohl im Handbuch steht, dass grundsätzlich die im InDesign angelegten Stilvorlagen die Definitionen der importierten überschreiben.

Der InDesign-RTF-Import unter Windows ist der mit Abstand leistungsfähigste. Das gleiche Programm unter MacOS zeigte bereits kleinere Fehler, die in der Windows-Version nicht zu Tage traten. Erstaunlicherweise verfügt Quark XPress in der Version für den Macintosh über keinen RTF-Import. Die Windows-Version konnte die erstellte RTF-Datei nicht importieren, da sie wahrscheinlich nur RTFs älterer Version konvertieren kann. (Da die Datei mit Word XP erstellt wurde, lag die Testdatei natürlich in der neuesten RTF-Version vor.) Hier besteht für XPress erhöhter Nachholbedarf. Für Quark XPress empfiehlt sich der Word-Import mit seinen beschriebenen Limitationen. Besonders bedauernswert ist hier die fehlende Unterstützung für Tabellen. Allgemein ist die Tabellenhandhabung von XPress sehr proprietär und inkompatibel, da Tabellen in eigenen Rahmen verwaltet werden und somit – ohne XTension-Support – keinerlei Möglichkeit besteht, diese direkt zu importieren. Ein Verankern im Text bedarf immer eines Cut-&-Paste-Vorganges: Der Tabellenrahmen wird ausgeschnitten und in den Textrahmen eingefügt.

Die Nutzung des Nur-Text-Formats kann als professioneller Weg ausgeschlossen werden. Sämtliche Formatierungen und Strukturierungen gehen beim Export aus Word verloren. Alles muss nachträglich per Hand neu ausgezeichnet werden – eine mühsame und zeitintensive Tätigkeit. Zumal bei den Versuchen der Nur-Text-

Exportfilter des Word Absatzeinzüge mit Leerzeichen simuliert hat, die alle wieder manuell entfernt werden müssen.

7.3 XML

Die Versuche haben aufgezeigt, dass die DTP-Applikationen derzeit weit davon entfernt sind, wirklich XML verarbeiten zu können. XML auf professionelle Art und Weise zu nutzen ist mit den derzeitigen Implementationen schlicht unmöglich.

Ist ein XML-Strom zu verarbeiten, empfiehlt sich unbedingt die Transformation über ein XSLT-oder DSSSL-Style-Sheet zu den proprietären Steuerzeichen, die DTP-Programme zur Verfügung stellen: XPress-Marken und Adobes Tagged Text. Nur so kann beispielsweise die Fußnotenproblematik akzeptabel gelöst werden, da keine der untersuchten DTP-Programme eine eigene Fußnotenverwaltung aufweist.

Auch diejenigen, die XML nur zur Formatierungs-Auszeichnung verwenden wollen, sollten – alleine schon der Verarbeitungsgeschwindigkeit wegen – eine Transformation der Daten vornehmen. Der Geschwindigkeitsverlust unter InDesign ist enorm.

Aus Word heraus XML zu generieren scheint somit unangebracht. Mit wenig komplexen XML-Strömen, bei denen die DTD eigens erstellt und die Instanzen per Hand codiert wurden (oder durch eine professionelle Applikation wie XMetal), kann die Verwendung von XML sinnvoll sein. Dies sollte aber unbedingt im Einzelfall abgeklärt und mit Versuchen ausgetestet werden.

7.4 Steuercodes der DTP-Programme

Die Konvertierung in Steuercodes erbringt die zuverlässigsten Ergebnisse. Je besser das VBA-Makro programmiert wurde, um so mehr Funktionalität kann von Word nach XPress oder InDesign übertragen werden. Einschränkungen ergeben sich nur im Leistungsumfang der XPress-Marken beziehungsweise des Tagged-Text-Formats. Hier hat Adobe eindeutig die bessere, weil umfangreichere Definition geliefert, da selbst Tabellen, Indices und Hyperlinks problemlos transportiert werden können.

7.5 Ergebnis

Es macht keinen Sinn eine allgemein zutreffende Aussage für alle Workflows treffen zu wollen. Die Arbeit hat gezeigt, dass die speziellen Konvertierungsfilter (für Word und RTF) ebenso unzureichend sind, wie die XML-Implementationen. Der Import von XPress-Marken oder Tagged Text liefert die saubersten und zuverlässigsten Ergebnisse. Welche Word-Funktionalitäten wie übertragen werden lässt sich hier leicht steuern und ermöglicht spezifische Lösungen für Einzelprobleme; es existiert also eine große Flexibilität.

Ob die Steuercodes über ein transformiertes XML-Dokument oder die VBA-Makro-Variante erstellt werden, ist nahezu unerheblich. Zu erwähnen bleibt, dass mit dem VBA-Makro ein Konvertierungsschritt – nämlich von Word zu XML – und damit eine weitere Fehlerquelle entfällt und direkt aus Word in die Steuercodes konvertiert werden kann.

Ob die Nutzung vorgefertigter Makro-Bibliotheken, wie dies beim mm²-Editor von Mazzetti & Mazzetti der Fall ist, oder die eigens programmierter VBA-Makros zu favorisieren ist, muss von Fall zu Fall entschieden werden. Dies hängt auch stark von der VBA-Kompetenz der Mitarbeiter ab. Doch sollte der Weg, das Makro selbst zu programmieren, unbedingt in Betracht gezogen werden. Er ermöglicht maximale Flexibilität. Mit der Zeit werden sich Codebibliotheken ansammeln, die mit vielfältigen Problemen fertig werden und typografische Feinheiten automatisch konvertieren können. Eine Anpassung auf spezielle Einsatzfelder (beispielsweise einen Datenbankimport) ist dann nur noch Detailarbeit.

In jedem Fall aber ist eines zu tun: Der festgelegte Weg, den ein Word-Dokument gehen soll, ist mit dem Autor genau abzusprechen. Sind alle Vorbereitungen getroffen –

also im Word sämtliche Formatvorlagen angelegt, alle Sonderzeichen und Sonderfälle geprüft und nicht benötigte Funktionen ausgeblendet – so muss dem Autor ein sauber vorbereitetes Word-Dokument zur Verfügung gestellt und erläutert werden. Es ist anzuraten sich dafür wirklich Zeit zu nehmen und eine Schulung mit dem Autor durchzuführen, damit er sich auf das Dokument einstellen kann und es richtig benutzt.

Bei der Anlage des Word-Dokuments, sollte der Autor auf einige grundlegende Dinge hingewiesen werden, die beim Umgang mit Word beachtet werden sollten, um eine maximale Verwendbarkeit der Daten zu erreichen.

Er sollte weder mit Tabulatoren, noch mit Leerzeichen versuchen, Texte bündig untereinander zu stellen. Nun muss dem Autor die Verwendung von Tabulatoren erklärt und mit ihm eingeübt werden, um unnötige Handarbeit zu vermeiden.

Ebenso verhält es sich mit harten Trennungen und Absatzmarken. Die Silbentrennung sollte generell ausgeschaltet sein, da Word in alle zu trennden Wörter weiche Trennungen einfügt, die unsinnigerweise mitkonvertiert werden. Harte Trennungen sind unbedingt zu unterlassen. Sie werden allzu leicht übersehen und ergeben ärgerliche Fehler im Endprodukt. Soll ein neuer Absatz erzeugt werden, dann muss auch die Return-Taste betätigt werden. Soll nur ein Umbruch stattfinden, ist ein weiches Absatzzeichen zu tasten (Shift-Return). In den VBA-Routinen sollten all diese Sonderzeichen im Word-Dokument gesucht und gegen das entsprechende Sonderzeichen für den XPress- oder InDesign-Import ersetzt werden.

Als Anführungszeichen verwendet der Autor am Besten nur das automatische (Shift-2) und tastet sie nicht direkt.

Er darf keinerlei Auszeichnungen per Hand benutzen, die nicht abgesprochen wurden. Jede dieser Auszeichnungen muss im Makro berücksichtigt werden. Die nicht berücksichtigten gehen nach dem Makrolauf unwiederbringlich verloren. Generell gilt der Grundsatz, dass der Autor alles per Formatvorlagen auszeichnen muss und nicht über die manuelle Zuweisung von Formatierung.

Grafiken und Bilder sollten nicht in Word erstellt und nicht eingebettet werden. Eine einfache Platzierung hilft dem Setzer bei der Konvertierungsarbeit. Sie ist auch unbedingt vonnöten, da sie den Stand der Grafiken und Bilder im Dokumentausdruck klar kennzeichnet.

Grundlegend gilt zu beachten, dass der Autor im Word nicht gestalten, sondern sich kleinlich an die Vorschriften halten sollte.

Nach getaner Arbeit muss das Word-Dokument sauber, das heißt ohne Schnellspeicherung, abgespeichert werden. Die Schnellspeicherung erzeugt ein word complex file, das von den DTP-Applikationen nicht oder nur fehlerhaft importiert werden kann.

Vor dem Verschicken muss ein Ausdruck des letzten gültigen Textstands erfolgen und dem Datenträger beigelegt werden. Dieser hilft bei Zuordnungsschwierigkeiten und ist zur Überprüfung der Konvertierung unerlässlich.

Kapitel 8

Ausblick

Diese Arbeit liefert viele Kritikpunkte an den Programmen, ihren Im- und Exportfiltern und der Art und Weise, wie verschiedene Funktionalitäten implementiert wurden. Doch in welche Richtung sollten sich die Programme weiter entwickeln?

Besieht man das Grundprinzip der Steuercodes für DTP-Programme, also XPress-Marken und Tagged Text, so kann ein direkter Vergleich zu SGML/XML gezogen werden. Adobe nennt die mit Steuercodes ausgezeichnete Datei bezeichnenderweise schon Tagged Text. Zu wünschen wäre eine XML- oder SGML-konforme Definition der Steuercodes. Eine solche Definition würde die Transformation aus einer XML- oder SGML-Instanz erheblich vereinfachen. Denn es muss die Frage gestellt werden, ob ein DTP-Programm wirklich direkt einen XML-Strom verarbeiten können muss.

Betrachtet man beispielsweise die HTML-Ausgabe von Quark XPress kritisch, ist diese nur bedingt geeignet im Web publiziert zu werden. Der generierte Code ist unter günstigen Bedingungen gut, doch in den meisten Fällen unzureichend. Gemeint ist damit mehreres: Die Umsetzung des Inhalts in unlogische und damit schlechte Strukturierung (was die Programmierer »Spagetticode« nennen), die fehlende oder eingeschränkte Anpassung an verschiedenste Browser und deren Versionen und die fehlende Bearbeitungsmöglichkeit von dynamischen Inhalten via PHP oder PERL und so weiter.

Und eben dies gilt auch für die untersuchten XML-Implementationen. Für den professionellen Einsatz sind sie völlig ungeeignet – Quarks Ansatz ist noch ungeeigneter als Adobes –, weshalb sich sofort die Frage stellt, ob so etwas überhaupt nötig ist.

Die Hersteller täten gut daran, auf die reibungslose Kommunikation zwischen ihren Programmen und XML zu achten, statt einem Programm, das von seiner Grundstruktur nicht dafür ausgelegt ist, eine solche Funktionalität nur unzufriedenstellend mitzugeben. XPress-Marken und InDesign-Tags XML-konform aufzubauen, wäre ein Schritt in die richtige Richtung. Auf diese Art und Weise würde sich beispielsweise auch der Export nach HTML um ein Vielfaches verbessern.

Ebenso lässt sich für Word argumentieren. Zwar besteht mit RTF schon ein guter Ansatz, doch wäre ein XML-konformes Dateiformat mit Sicherheit eine große Erleichterung für die Konvertierungsarbeit. Zwei XML-konforme Formate ineinander zu transformieren ist um vieles leichter, als zwei proprietäre Formate. Doch besteht hier wieder die Gefahr, dass Microsoft versucht, den offenen XML-Standard mit Microsoft-eigenen Funktionalitäten zu untergraben, wie beispielsweise bei Java und HTML geschehen.

In diesem Zuge sollten die Steuercodes der DTP-Programme noch weiter ausgebaut werden, um möglichst viel Programmfunktionalität automatisch aus den Steuercodes ansteuern zu können. Natürlich ergänzt eine gute Skriptfähigkeit, per VBA oder Apple Script, dies ideal, da somit auch direkte Interaktionen mit anderen Programmen und Manipulationen in einem größeren Rahmen als lediglich auf der Textebene möglich sind.

Während InDesign schon recht nahe an diese Wunschvorstellung herankommt, ist XPress weit davon entfernt. Allein das Tabellenmodul sträubt sich gegen jegliche Codierung und wird unter Beibehaltung des derzeitigen Aufbaus auch nur unzureichend oder gar nicht in ein XML-orientiertes Konzept einzubringen sein.

Um noch zwei weitere Beispiele zu nennen: Die Definitionen von Hyperlinks und Indices hat noch nicht genug Einzug in die XPress-Marken gefunden. XPress stellt hier weit weniger Funktionalität zur Verfügung als InDesign.

Wünschenswert wäre auch eine durchgängige Unterstützung der verschiedenen Unicode-Encodings. Während Word und InDesign schon voll Unicode-basiert arbeiten, hängt Quark XPress noch immer zu nahe an den alten Systemencodings Macintosh-Standard und windows-1252.

DTP-Programme werben verstärkt mit Schlagworten wie Medienneutralität, Ausgabe auf den unterschiedlichsten Medien, HTML-Export und so weiter. Natürlich sind dies auch Forderungen, die der Markt stellt, und die Hersteller wollen eine möglichst große Zielgruppe bedienen. Doch statt auf den Kerngebieten die Kompetenz auszuweiten, versuchen die Anbieter immer neue Wege zu beschreiten. Das Ergebnis sind – wie diese Arbeit zeigt – meist halbherzige Versuche, die alle an der Produktionskette Beteiligten nicht zufrieden stellen beziehungsweise verärgern. Oft müssen faule Kompromisse geschlossen oder Umwege (Workarounds) gegangen werden, obwohl man mit einer besseren und intelligenteren Implementation direkter zum Ziel kommen könnte.

So besteht für die Hersteller noch viel Handlungsbedarf bei der Applikationsentwicklung, sowohl im Detail als auch auf konzeptioneller Ebene.

Kapitel 9

Anhang

9.1 Übersicht über ISO 8859

Hier eine kurze Übersicht der gebräuchlichsten ISO-8859-x-Encodings. Die Encoding-Tabellen sind frei im Internet verfügbar.

- *ISO 8859-1 (Latin-1):* für Englisch, Dänisch, Niederländisch, Finnisch, Französisch, Deutsch, Isländisch, Irisch, Italienisch, Bahasa Malaysisch, Norwegisch, Portugiesisch, Spanisch, Schwedisch, Tagalog (Philippinen).
- *ISO 8859-2 (Latin-2):* für Albanisch, Tschechisch, Deutsch, Ungarisch, Polnisch, Rumänisch, Kroatisch, Slowakisch, Slowenisch, Schwedisch.
- *ISO 8859-3 (Latin-3):* für Afrikaans, Katalanisch, Französisch, Galizisch, Italienisch, Maltesisch, Türkisch.
- *ISO 8859-4 (Latin-4):* für Dänisch, Estnisch, Finnisch, Deutsch, Grönländisch, Lappisch, Litauisch, Norwegisch, Schwedisch. (Jetzt ersetzt durch ISO 8859-10: Latin-6).
- *ISO 8859-5 (Latin/Cyrillic):* Bulgarisch, Belorussisch, Mazedonisch, Russisch, Serbisch, Ukrainisch.
- *ISO 8859-6 (Latin/Arabic):* Arabisch ohne Punktation.
- *ISO 8859-7 (Latin/Greek):* Griechisch.
- *ISO 8859-8 (Latin/Hebrew):* Hebräisch ohne Punktation.
- *ISO 8859-9 (Latin-5):* Finnisch, Französisch, Deutsch, Irisch, Italienisch, Norwegisch, Portugiesisch, Spanisch, Schwedisch, Türkisch.
- *ISO 8859-10 (Latin-6):* Dänisch, Estnisch, Faroerisch, Finnisch, Deutsch, Grönländisch (Inuit), Isländisch, Lappisch (Samisch), Litauisch, Lettisch, Norwegisch, Schwedisch.
- *ISO 8859-14 (Latin-7):* Gälische Sprachen.
- *ISO 8859-15 (Latin-9):* Latin-1 mit Euro.

9.2 CODIERUNG DER SONDERZEICHEN UNTER MACOS 9.2.1

Wie kompliziert es ist, ein Format in ein anderes zu überführen, zeigt die nachfolgende Tabelle recht gut. Sie stellt die Codierung der verwendbaren Sonderzeichen[54] in den verschiedenen Dateiformaten unter MacOS 9.2.1 dar. Alleine schon diese Konvertierung stellt einen Entwickler vor diffizile Programmierarbeit.

[alle Werte in Hex-Code]	InDesign	XPress	Word
Sonderzeichen			
Automatische Seitenzahl	18[55]	04[55]	13 20 50 [PAGE] 20 20 [*] 20 [MERGEFORMAT] 20 14 [1] 15[56]
Nächste Seitenzahl	18[55]	04[55]	–[56]
Vorherige Seitenzahl	18[55]	04[55]	–[56]
Abschnittsname	19	–	–[56]
Bedingter Trennstrich	AD	1F	1F
Geschützter Trennstrich	80 01 20 11 40 05	2D	1E
Tabulator zum rechten Rand	08	09	–
Einzug hier	07	1E	–
Leerräume			
Geviert	80 01 20 03 40 05	–	–
Halbgeviert	80 01 20 02 40 05	CA	–
Ausgleichsabstand	80 01 20 01 40 05	–	–
Achtelgeviert	80 01 20 0A 40 05	–	–
Geschütztes Leerzeichen	A0	20[55]	A0
Viertelgeviert	80 01 20 09 40 05	0F[55]	–
Raum für Zahlen	80 01 20 07 40 05	–	–
Raum für Satzzeichen	80 01 20 08 40 05	10	–
Flexibles Leerzeichen	–	0F[55]	–
Umbruchzeichen			
Spaltenumbruch	0D[55]	0B	0E
Rahmenumbruch	0D[55]	0C	–
Seitenumbruch	0D[55]	–	0C
Umbruch für gerade Seiten	0D[55]	–	–[56]
Umbruch für ungerade Seiten	0D[55]	–	–[56]
Weicher Zeilenumbruch	0A	07	0B
Absatzumbruch	0D[55]	0D	0D

54 Gemeint sind hier Programmfunktionen, nicht etwa Zeichen im Sinne des Kapitels 4.1.

55 Das Programm erkennt beispielsweise einen Umbruchbefehl (0D) und durchsucht eine Look-up-Tabelle (LUT), um die Art des Umbruchs näher zu spezifizieren: ob Spalten-, Rahmenumbruch und so weiter. In der LUT ist der Byte-Offset des Zeichens (vom Beginn der Datei ab) und ein Steuercode, der die Art weiter definiert, gespeichert.

56 Diese Funktionalität ist in Word über die Feldfunktionen erreichbar. Die Feldfunktionen haben folgende Syntax: `13 20 [function 20 [...]] 20 [MERGEFORMAT] 20 14 [result] 15`. `[result]` beinhaltet das Ergebnis der vorher definierten `[function]`.
Über die Feldfunktionen lassen sich auch Abfragen mit Wenn-dann-Bedingungen realisieren.

9.3 BACKUS-NAUR-FORM

Ist eine Sprache in der Backus-Naur-Form (BNF) festgehalten und definiert, kann sie leicht mittels eines Algorithmus auf Richtigkeit überprüft werden. Die BNF bildet die Grundlage von Compilern und Parsern in den verschiedenen Programmier- und Metasprachen wie C, Java, Basic und XML/SGML. Sie wurde in den 1950er Jahren für die Programmiersprache Algol60 entwickelt.

Ein komplexes syntaktisches Geflecht wird in kleinere Einheiten aufgeteilt und mit einem festen Regelwerk beschrieben. Ein einfaches Beispiel, das nichts mit XML, SGML oder realen Sprachstrukturen zu tun hat, soll die doch recht schwierig zu beschreibende Begrifflichkeit fassbarer machen:

```
[1] Wort       ::= Konsonant Vokal+
                   Konsonant
[2] Konsonant  ::= [^aeiou]
[3] Vokal      ::= [aeiou]
```

Regel Nummer 1 definiert (`::=`), dass die syntaktische Einheit »Wort« aus dem Folgenden besteht: einem Konsonant, mindestens einem oder mehreren Vokalen und abschließend wieder einem Konsonanten. Die Reihenfolge ist hierbei streng einzuhalten. Regel 2 definiert einen Konsonanten als einen Buchstaben, der nicht (^) a, e, i, o oder u ist. Dem entgegen definiert Regel 3 einen Vokal als einen Buchstaben aus der Menge a, e, i, o oder u.

Betrachten wir – wie ein Compiler – schrittweise, ob »rot« ein Wort im Sinne unserer Definition ist: »Rot« ist »r« gefolgt von »o« gefolgt von »t«. »r« ist ein Konsonant nach Regel 2, »o« ein Vokal nach Regel 3 und »t« ein Konsonant nach Regel 2, so dass sich die Abfolge ergibt: Konsonant, Vokal, Konsonant. Dies wiederum entspricht unserer Regel 1.

So trivial die Regelaufstellung und -befolgung bei diesem Beispiel wirken mag, so kompliziert wird sie bei der Definition und Überprüfung einer realen Sprachstruktur.

Wie schon erwähnt, basiert die komplette Definition der Meta-Sprache XML auf der BNF (zu sehen unter http://www.w3.org/TR/2000/REC-xml-20001006, Stand: 4. 9. 2002), aber auch das Rich Text Format wurde BNF-konform definiert (vergleiche Kapitel 4.3.1.1). Einige Elemente der BNF-Notation sollen hier dargelegt werden, da sie für die Sprachedefinition mit XML ebenfalls zur Anwendung kommen:

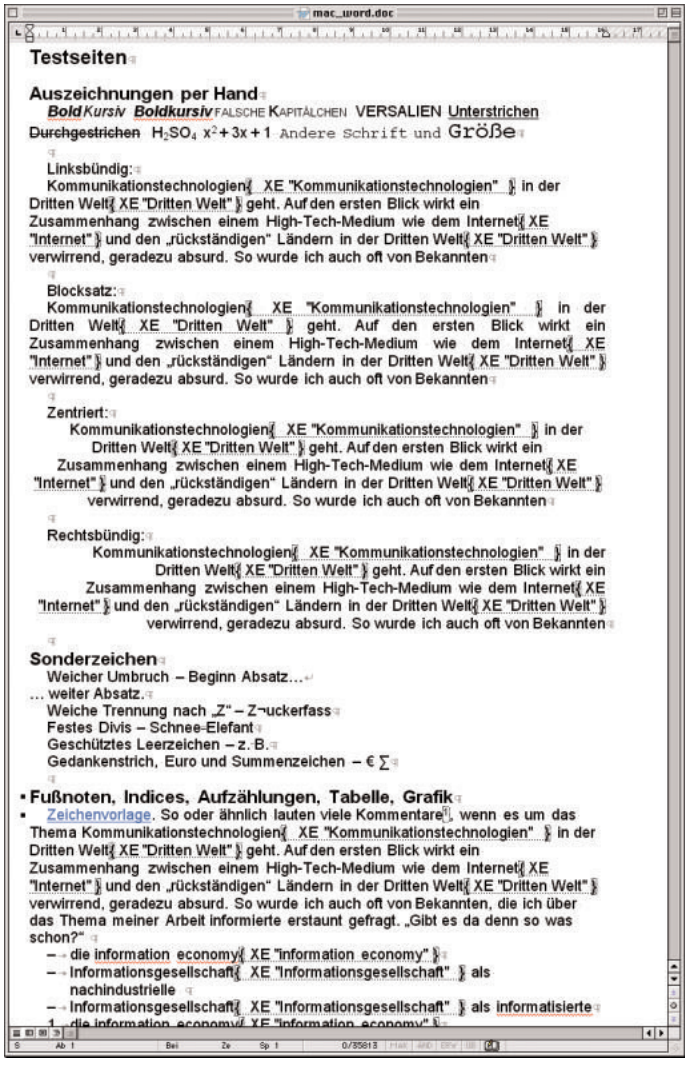
`A B`	entspricht `A` gefolgt von `B`
`A | B`	entspricht `A` oder `B`
`A?`	entspricht `A` oder nichts
`A+`	entspricht mindestens ein- oder mehrmals `A`
`A*`	entspricht kein-, ein- oder mehrmals `A`

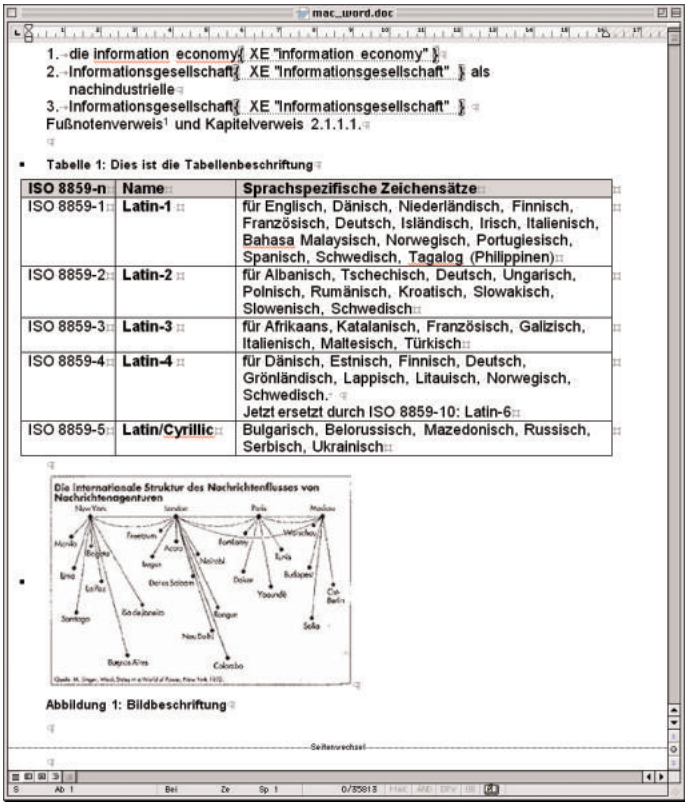
9.4 XML-konformer Teil einer InDesign-Datei

Der nachfolgende Bereich ist einer InDesign-Datei entnommen. Er ist Teil der Druck-Definitionen (plist für print-list):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "file://localhost/System/Library/DTDs/
                                                    PropertyList.dtd">
<plist version="0.9">
<dict>
   <key>com.apple.print.PageFormat.BackupPrintRecord</key>
   <dict>
      <key>com.apple.print.ticket.creator</key>
      <string>com.apple.printingmanager</string>
      <key>com.apple.print.ticket.itemArray</key>
      <array>
         <dict>
            <key>com.apple.print.PageFormat.BackupPrintRecord</key>
            <data>
            ACgAAABIAEgAAAAAxgCQf/3//cDQAJKIAIFewPgAAAA
            AAFoAWgAAAAAD3gLRQFsADILRUcYAFAAAQEBAAAAAScP
            AAEAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAZAAAAAAA
            AAAAAgAQsQUAERAAAEUAAAA2TVYg
            </data>
            <key>com.apple.print.ticket.client</key>
            <string>com.apple.printingmanager</string>
            <key>com.apple.print.ticket.modDate</key>
            <date>2002-04-11T15:45:28Z</date>
            <key>com.apple.print.ticket.stateFlag</key>
            <integer>0</integer>
         </dict>
      </array>
   </dict>
<!— hier wurde Code gekürzt —>
</dict>
</plist>
```

9.5.1
Als Worddatei





9.5.2
XML-codiert

```
<?xml version="1.0" encoding="UTF-8" ?>
<!–
This document was converted from RTF source by upCast Private Edition (c)
1999-2001 <www.infinity-loop.de>
–> <document>
<documentinfo>
  <property name="title" value="Dritte Welt und der Aufbruch in das
           Informationszeitalter &#8211; am Beispiel Internet" />
  <property name="author" value="mr holmes" />
  <property name="operator" value="mr holmes" />
  <property name="creationTime" value="Mon, 21 Oct 2002 18:28:00 CEST" />
  <property name="revisionTime" value="Mon, 21 Oct 2002 18:28:00 CEST" />
  <property name="editingMinutes" value="0" />
  <property name="numberOfPages" value="1" />
  <property name="numberOfWords" value="40540" />
  <property name="numberOfChars" value="255407" />
  <property name="company" value="221b baker street" />
  <property name="numberOfCharsWS" value="295357" />
</documentinfo>

<part>
  <section level="1">
    <heading level="1" name="Title">Testseiten</heading>
    <Normal>Auszeichnungen per Hand</Normal>
    <Normal>Bold Kursiv  Boldkursiv falsche Kapit&#228;lchen  Versalien
           Unterstrichen  Durchgestrichen  H<subscript>2</subscript>
           SO<subscript>4</subscript>  x<superscript>2</superscript>
           + 3x + 1  Andere Schrift und Gr&#246;&#223;e</Normal>
    <Normal></Normal>
    <Normal>Linksb&#252;ndig:</Normal>
    <Normal>Kommunikationstechnologien<hidden>
        <indextarget main="Kommunikationstechnologien" sub="" /></hidden>
           in der Dritten Welt<hidden>
        <indextarget main="Dritten Welt" sub="" /></hidden> geht. Auf den
           ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-
           Medium wie dem Internet<hidden>
        <indextarget main="Internet" sub="" /></hidden> und den
           &#8222;r&#252;ckst&#228;ndigen&#8220; L&#228;ndern in der
           Dritten Welt<hidden>
```

```
    <indextarget main="Dritten Welt" sub="" /></hidden> verwirrend,
        geradezu absurd. So wurde ich auch oft von Bekannten</Normal>
<Normal></Normal>
<Normal>Blocksatz:</Normal>
<Normal>Kommunikationstechnologien<hidden>
    <indextarget main="Kommunikationstechnologien" sub="" /></hidden>
        in der Dritten Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> geht. Auf den
        ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-
        Medium wie dem Internet<hidden>
    <indextarget main="Internet" sub="" /></hidden> und den
        &#8222;r&#252;ckst&#228;ndigen&#8220; L&#228;ndern in der
        Dritten Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> verwirrend,
        geradezu absurd. So wurde ich auch oft von Bekannten</Normal>
<Normal></Normal>
<Normal>Zentriert:</Normal>
<Normal>Kommunikationstechnologien<hidden>
    <indextarget main="Kommunikationstechnologien" sub="" /></hidden>
        in der Dritten Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> geht. Auf den
        ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-
        Medium wie dem Internet<hidden>
    <indextarget main="Internet" sub="" /></hidden> und den
        &#8222;r&#252;ckst&#228;ndigen&#8220; L&#228;ndern in der
        Dritten Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> verwirrend,
        geradezu absurd. So wurde ich auch oft von Bekannten</Normal>
<Normal></Normal>
<Normal>Rechtsb&#252;ndig:</Normal>
<Normal>Kommunikationstechnologien<hidden>
    <indextarget main="Kommunikationstechnologien" sub="" /></hidden>
        in der Dritten Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> geht. Auf den
        ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-
        Medium wie dem Internet<hidden>
    <indextarget main="Internet" sub="" /></hidden> und den
        &#8222;r&#252;ckst&#228;ndigen&#8220; L&#228;ndern in der Dritten
        Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> verwirrend,
        geradezu absurd. So wurde ich auch oft von Bekannten</Normal>
<Normal></Normal>
```

```
<Normal>Sonderzeichen</Normal>
<Normal>Weicher Umbruch &#8211; Beginn Absatz&#8230;<newline />&#8230;
        weiter Absatz.</Normal>
<Normal>Weiche Trennung nach &#8222;Z&#8220; &#8211; Zuckerfass
        </Normal>
<Normal>Festes Divis &#8211; Schnee&#8209;Elefant</Normal>
<Normal>Gesch&#252;tztes Leerzeichen &#8211; z. B.</Normal>
<Normal>Gedankenstrich, Euro und Summenzeichen &#8211; &#8364;
        &#8721;</Normal>
<Normal></Normal>
<Normal>Fu&#223;noten, Indices, Aufz&#228;hlungen, Tabelle, Grafik
        </Normal>
<Normal><Hyperlink>Zeichenvorlage</Hyperlink>. So oder &#228;hnlich
        lauten viele Kommentare<superscript><footnote_reference>
      <footnote>
        <footnote_text><target name="_Ref22528779" />
        Fu&#223;notentest!</footnote_text></footnote>
        </footnote_reference></superscript>, wenn es um das Thema
        Kommunikationstechnologien<hidden>
    <indextarget main="Kommunikationstechnologien" sub="" /></hidden>
        in der Dritten Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> geht. Auf den
        ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-
        Medium wie dem Internet<hidden>
    <indextarget main="Internet" sub="" /></hidden> und den
        &#8222;r&#252;ckst&#228;ndigen&#8220; L&#228;ndern in der
        Dritten Welt<hidden>
    <indextarget main="Dritten Welt" sub="" /></hidden> verwirrend,
        geradezu absurd. So wurde ich auch oft von Bekannten, die
        ich &#252;ber das Thema meiner Arbeit informierte erstaunt
        gefragt. &#8222;Gibt es da denn so was schon?&#8220; </Normal>
<list listtype="disc">
  <item>
    <Normal>die information economy<hidden>
        <indextarget main="information economy" sub="" /></hidden>
        </Normal>
  </item>
  <item>
    <Normal>Informationsgesellschaft<hidden>
        <indextarget main="Informationsgesellschaft" sub="" />
        </hidden> als nachindustrielle </Normal>
  </item>
```

```
  <item>
    <Normal>Informationsgesellschaft<hidden>
        <indextarget main="Informationsgesellschaft" sub="" />
        </hidden> als informatisierte</Normal>
  </item>
</list>
<list listtype="ordinal">
  <item>
    <Normal>die information economy<hidden>
        <indextarget main="information economy" sub="" /></hidden>
        </Normal>
  </item>
  <item>
    <Normal>Informationsgesellschaft<hidden>
        <indextarget main="Informationsgesellschaft" sub="" />
        </hidden> als nachindustrielle</Normal>
  </item>
  <item>
    <Normal>Informationsgesellschaft<hidden>
        <indextarget main="Informationsgesellschaft" sub="" />
        </hidden> </Normal>
  </item>
</list>
<Normal>Fu&#223;notenverweis<reference name="_Ref22528779">
        <superscript><footnote_reference>1</footnote_reference>
        </superscript></reference> und Kapitelverweis
        <reference name="2.1.1.1">2.1.1.1</reference>.
        </Normal>
<Normal></Normal>
<caption>Tabelle 1: Dies ist die Tabellenbeschriftung</caption>
<table>
  <row>
    <cell>
      <Normal>ISO 8859-n</Normal>
    </cell>
    <cell>
      <Normal>Name</Normal>
    </cell>
    <cell>
      <Normal>Sprachspezifische Zeichens&#228;tze</Normal>
    </cell>
  </row>
```

```
<row>
  <cell>
    <Normal>ISO 8859-1</Normal>
  </cell>
  <cell>
    <Normal>Latin-1 </Normal>
  </cell>
  <cell>
    <Normal>f&#252;r Englisch, D&#228;nisch, Niederl&#228;ndisch,
            Finnisch, Franz&#246;sisch, Deutsch, Isl&#228;ndisch,
            Irisch, Italienisch, Bahasa Malaysisch, Norwegisch,
            Portugiesisch, Spanisch, Schwedisch, Tagalog (Philippinen)
            </Normal>
  </cell>
</row>
<row>
  <cell>
    <Normal>ISO 8859-2</Normal>
  </cell>
  <cell>
    <Normal>Latin-2 </Normal>
  </cell>
  <cell>
    <Normal>f&#252;r Albanisch, Tschechisch, Deutsch, Ungarisch,
            Polnisch, Rum&#228;nisch, Kroatisch, Slowakisch,
            Slowenisch, Schwedisch</Normal>
  </cell>
</row>
<row>
  <cell>
    <Normal>ISO 8859-3</Normal>
  </cell>
  <cell>
    <Normal>Latin-3 </Normal>
  </cell>
  <cell>
    <Normal>f&#252;r Afrikaans, Katalanisch, Franz&#246;sisch,
            Galizisch, Italienisch, Maltesisch, T&#252;rkisch
            </Normal>
  </cell>
</row>
<row>
```

```
        <cell>
          <Normal>ISO 8859-4</Normal>
        </cell>
        <cell>
          <Normal>Latin-4 </Normal>
        </cell>
        <cell>
          <Normal>f&#252;r D&#228;nisch, Estnisch, Finnisch, Deutsch,
          Gr&#246;nl&#228;ndisch, Lappisch, Litauisch, Norwegisch,
          Schwedisch.  </Normal>
          <Normal>Jetzt ersetzt durch ISO 8859-10: Latin-6</Normal>
        </cell>
      </row>
      <row>
        <cell>
          <Normal>ISO 8859-5</Normal>
        </cell>
        <cell>
          <Normal>Latin/Cyrillic</Normal>
        </cell>
        <cell>
          <Normal>Bulgarisch, Belorussisch, Mazedonisch, Russisch, Serbisch,
          Ukrainisch</Normal>
        </cell>
      </row>
    </table>
    <Normal></Normal>
    <Normal>
      <image src="D:\arbeit\dtobo\testparcour_pc\konvertiertetestdateien\
      dip_img\nachricht.jpg" /></Normal>
    <caption>Abbildung 1: Bildbeschriftung</caption>
    <Normal></Normal><pagebreak />
    <Normal></Normal>
  </section>

[...]
```

9.5.3
XPress-Marken-codiert

```
@titel:Testseiten
@standard:Auszeichnungen per Hand
@standard:Bold Kursiv  Boldkursiv FALSCHE KAPITÄLCHEN  VERSALIEN
Unterstrichen  Durchgestrichen  H2SO4  x2 + 3x + 1  Andere Schrift
und Größe
@standard:
@standard:Linksbündig:
@standard:Kommunikationstechnologien in der Dritten Welt geht. Auf
den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium
wie dem Internet und den "rückständigen" Ländern in der Dritten Welt
verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten
@standard:
@standard:Blocksatz:
@standard:Kommunikationstechnologien in der Dritten Welt geht. Auf
den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium
wie dem Internet und den "rückständigen" Ländern in der Dritten Welt
verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten
@standard:
@standard:Zentriert:
@standard:Kommunikationstechnologien in der Dritten Welt geht. Auf
den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium
wie dem Internet und den "rückständigen" Ländern in der Dritten Welt
verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten
@standard:
@standard:Rechtsbündig:
@standard:Kommunikationstechnologien in der Dritten Welt geht. Auf
den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium
wie dem Internet und den "rückständigen" Ländern in der Dritten Welt
verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten
@standard:
@standard:Sonderzeichen
@standard:Weicher Umbruch – Beginn Absatz...
... weiter Absatz.
@standard:Weiche Trennung nach "Z" – Zuckerfass
@standard:Festes Divis – SchneeElefant
@standard:Geschütztes Leerzeichen – z. B.
@standard:Gedankenstrich, Euro und Summenzeichen – € ∑
@standard:
@standard:Fußnoten, Indices, Aufzählungen, Tabelle, Grafik
```

@standard:<@hyperlink>Zeichenvorlage<@$p>. So oder ähnlich lauten viele Kommentare<@fußnotenzeichen>1<@$p>, wenn es um das Thema Kommunikationstechnologien in der Dritten Welt geht. Auf den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium wie dem Internet und den "rückständigen" Ländern in der Dritten Welt verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten, die ich über das Thema meiner Arbeit informierte erstaunt gefragt. "Gibt es da denn so was schon?"
@standard:- die information economy
@standard:- Informationsgesellschaft als nachindustrielle
@standard:- Informationsgesellschaft als informatisierte
@standard:1. die information economy
@standard:2. Informationsgesellschaft als nachindustrielle
@standard:3. Informationsgesellschaft
@standard:Fußnotenverweis<@fußnotenzeichen>1<@$p> und Kapitelverweis 2.1.1.1.
@standard:
@beschriftung:Tabelle 1: Dies ist die Tabellenbeschriftung
@standard:ISO 8859-n Name Sprachspezifische Zeichensätze
@standard:ISO 8859-1 Latin-1 für Englisch, Dänisch, Niederländisch, Finnisch, Französisch, Deutsch, Isländisch, Irisch, Italienisch, Bahasa Malaysisch, Norwegisch, Portugiesisch, Spanisch, Schwedisch, Tagalog (Philippinen)
@standard:ISO 8859-2 Latin-2 für Albanisch, Tschechisch, Deutsch, Ungarisch, Polnisch, Rumänisch, Kroatisch, Slowakisch, Slowenisch, Schwedisch
@standard:ISO 8859-3 Latin-3 für Afrikaans, Katalanisch, Französisch, Galizisch, Italienisch, Maltesisch, Türkisch
@standard:ISO 8859-4 Latin-4 für Dänisch, Estnisch, Finnisch, Deutsch, Grönländisch, Lappisch, Litauisch, Norwegisch, Schwedisch.
@standard:Jetzt ersetzt durch ISO 8859-10: Latin-6
@standard:ISO 8859-5 Latin/Cyrillic Bulgarisch, Belorussisch, Mazedonisch, Russisch, Serbisch, Ukrainisch
@standard:
@standard:
@beschriftung:Abbildung 1: Bildbeschriftung
@standard:
@standard:

9.5.4
Tagged-Text-codiert

```
<ASCII-WIN>
<DefineCharStyle:fett>
<DefineCharStyle:fußnotenzeichen>
<DefineCharStyle:hyperlink>
<DefineCharStyle:ü2_char>
<DefineCharStyle:ü4_char>
<DefineCharStyle:zitat>
<DefineParaStyle:titel>
<DefineParaStyle:beschriftung>
<DefineParaStyle:bildunterschrift>
<DefineParaStyle:fußnotentext>
<DefineParaStyle:standard>
<DefineParaStyle:ü1>
<DefineParaStyle:ü2>
<DefineParaStyle:ü3>
<DefineParaStyle:ü4>
<DefineParaStyle:ü5>
<DefineParaStyle:untertitel>
<DefineParaStyle:zitat>
<ParaStyle:titel>Testseiten
<ParaStyle:standard>Auszeichnungen per Hand
<ParaStyle:standard>Bold Kursiv  Boldkursiv FALSCHE KAPITÄLCHEN
VERSALIEN  Unterstrichen  Durchgestrichen  H2SO4  x2 + 3x + 1
Andere Schrift und Größe
<ParaStyle:standard>
<ParaStyle:standard>Linksbündig:
<ParaStyle:standard>Kommunikationstechnologien in der Dritten Welt
geht. Auf den ersten Blick wirkt ein Zusammenhang zwischen einem
High-Tech-Medium wie dem Internet und den "rückständigen" Ländern in
der Dritten Welt verwirrend, geradezu absurd. So wurde ich auch oft
von Bekannten
<ParaStyle:standard>
<ParaStyle:standard>Blocksatz:
<ParaStyle:standard>Kommunikationstechnologien in der Dritten Welt
geht. Auf den ersten Blick wirkt ein Zusammenhang zwischen einem
High-Tech-Medium wie dem Internet und den "rückständigen" Ländern in
der Dritten Welt verwirrend, geradezu absurd. So wurde ich auch oft
von Bekannten
<ParaStyle:standard>
```

<ParaStyle:standard>Zentriert:
<ParaStyle:standard>Kommunikationstechnologien in der Dritten Welt geht. Auf den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium wie dem Internet und den "rückständigen" Ländern in der Dritten Welt verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten
<ParaStyle:standard>
<ParaStyle:standard>Rechtsbündig:
<ParaStyle:standard>Kommunikationstechnologien in der Dritten Welt geht. Auf den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium wie dem Internet und den "rückständigen" Ländern in der Dritten Welt verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten
<ParaStyle:standard>
<ParaStyle:standard>Sonderzeichen
<ParaStyle:standard>Weicher Umbruch – Beginn Absatz...
... weiter Absatz.
<ParaStyle:standard>Weiche Trennung nach "Z" – Zuckerfass
<ParaStyle:standard>Festes Divis – SchneeElefant
<ParaStyle:standard>Geschütztes Leerzeichen – z. B.
<ParaStyle:standard>Gedankenstrich, Euro und Summenzeichen – € ∑
<ParaStyle:standard>
<ParaStyle:standard>Fußnoten, Indices, Aufzählungen, Tabelle, Grafik
<ParaStyle:standard><CharStyle:hyperlink>Zeichenvorlage<CharStyle:>. So oder ähnlich lauten viele Kommentare<CharStyle:fußnotenzeichen>1 <CharStyle:>, wenn es um das Thema Kommunikationstechnologien in der Dritten Welt geht. Auf den ersten Blick wirkt ein Zusammenhang zwischen einem High-Tech-Medium wie dem Internet und den "rückständigen" Ländern in der Dritten Welt verwirrend, geradezu absurd. So wurde ich auch oft von Bekannten, die ich über das Thema meiner Arbeit informierte erstaunt gefragt. "Gibt es da denn so was schon?"
<ParaStyle:standard>-   die information economy
<ParaStyle:standard>-   Informationsgesellschaft als nachindustrielle
<ParaStyle:standard>-   Informationsgesellschaft als informatisierte
<ParaStyle:standard>1.   die information economy
<ParaStyle:standard>2.   Informationsgesellschaft als nachindustrielle
<ParaStyle:standard>3.   Informationsgesellschaft
<ParaStyle:standard>Fußnotenverweis<CharStyle:fußnotenzeichen>1 <CharStyle:> und Kapitelverweis 2.1.1.1.
<ParaStyle:standard>
<ParaStyle:beschriftung>Tabelle 1: Dies ist die Tabellenbeschriftung
<ParaStyle:standard>ISO 8859-n Name Sprachspezifische Zeichensätze

<ParaStyle:standard>ISO 8859-1 Latin-1 für Englisch, Dänisch, Niederländisch, Finnisch, Französisch, Deutsch, Isländisch, Irisch, Italienisch, Bahasa Malaysisch, Norwegisch, Portugiesisch, Spanisch, Schwedisch, Tagalog (Philippinen)
<ParaStyle:standard>ISO 8859-2 Latin-2 für Albanisch, Tschechisch, Deutsch, Ungarisch, Polnisch, Rumänisch, Kroatisch, Slowakisch, Slowenisch, Schwedisch
<ParaStyle:standard>ISO 8859-3 Latin-3 für Afrikaans, Katalanisch, Französisch, Galizisch, Italienisch, Maltesisch, Türkisch
<ParaStyle:standard>ISO 8859-4 Latin-4 für Dänisch, Estnisch, Finnisch, Deutsch, Grönländisch, Lappisch, Litauisch, Norwegisch, Schwedisch.
<ParaStyle:standard>Jetzt ersetzt durch ISO 8859-10: Latin-6
<ParaStyle:standard>ISO 8859-5 Latin/Cyrillic Bulgarisch, Belorussisch, Mazedonisch, Russisch, Serbisch, Ukrainisch
<ParaStyle:standard>
<ParaStyle:standard>
<ParaStyle:beschriftung>Abbildung 1: Bildbeschriftung
<ParaStyle:standard>
<ParaStyle:standard>

9.6 QUELLTEXT DES BEISPIELMAKROS TAGNOW

Das Makro wurde für Word XP auf Windows optimiert.
Es ist allerdings mit Office X und Office 2001 auf dem
MacOS uneingeschränkt lauffähig.

```
' tagNow() Makro
' tobias@wantzen.com
'
' diplomarbeit 2002, hochschule der medien, stuttgart.
'
'
' demo-makro zur auszeichnung von word-dokumenten.
' letzte bearbeitung: 12.11.2002

Public Sub tagNow()

'zeigt dialog, um seq. datei zu laden, die tags enthält.
'
Dim DatName As String
Dim Button As Integer
With Application.Dialogs(wdDialogFileOpen)
        Button = .Display
        DateiName = .Name
End With
If Button = 0 Then End  'wenn abbrechen gedrückt: ende.


'speichert die anzahl der zeilen der seq. datei in der variablen n.
'
Open DateiName For Input As #1
n = 0
Do While Not EOF(1)  'loop until end of file.
        Line Input #1, TextLine
        n = n + 1   'zählt durchläufe mit.
Loop
Close #1
```

```
'dimensionierung des formatvorlagen-arrays (fv-arrays) und
'definition einiger konstanten:
'conName  = array-postition des fv-namens,
'conStart = array-position des start-tags,
'conEnd   = array-position des ende-tags,
'conArt   = array-position der absatz-/zeichenorientiert-info.
ReDim FVarray(n, 3) As String
Const conName As Integer = 0, conStart As Integer = 1, _
      conEnd As Integer = 2, conArt As Integer = 3


'auslesen der fv und tags aus der seq. datei und füllen des fv-arrays.
'
Open DateiName For Input As #1
t = 0
Do While Not EOF(1) 'loop until end of file.
        Line Input #1, TextLine

        'dann trennen nach delimiter <tab> und dem array zuweisen.
        LetzterTab = Len(TextLine) - 1
        myHelp = 0
        For h = 3 To 0 Step -1
            myHelp = LetzterTab
            Do While Mid(TextLine, LetzterTab, 1) <> Chr$(9)
                LetzterTab = LetzterTab - 1
                If LetzterTab = 0 Then Exit Do
            Loop
            If h = 3 Then
                FVarray(t, h) = Mid(TextLine, LetzterTab + 1, myHelp - _
                                    LetzterTab + 1)
            Else
                FVarray(t, h) = Mid(TextLine, LetzterTab + 1, myHelp - _
                                    LetzterTab)
            End If
            LetzterTab = LetzterTab - 1
        Next h
        t = t + 1
Loop
Close #1
```

```
'alle tabellen eines dokuments in text umwandeln (tabs getrennt).
'
If ActiveDocument.Tables.Count >= 1 Then
    For Each myTable In ActiveDocument.Tables
        ActiveDocument.Tables(1).ConvertToText Separator:=wdSeparateByTabs
    Next myTable
End If


'gliederungen und aufzählungen aufheben
'
If ActiveDocument.Lists.Count >= 1 Then
    For Each myList In ActiveDocument.Lists
        ActiveDocument.Lists(1).ConvertNumbersToText
    Next myList
End If


'taggen des word-dokuments.
'
For h = 0 To n - 1      'zeichenorientierte fv
    If FVarray(h, conArt) = "0" Then _
        tagZeichen FVarray(h, conName), FVarray(h, conStart), _
            FVarray(h, conEnd)
Next h

For h = 0 To n - 1      'absatzorientierte fv
    If FVarray(h, conArt) = "1" Then _
        tagAbsatz FVarray(h, conName), FVarray(h, conStart), _
            FVarray(h, conEnd)
Next h
```

```
'alle absatzeinzüge auf 0 setzen
'
    Selection.WholeStory
    With Selection.ParagraphFormat
        .LeftIndent = MillimetersToPoints(0)
        .SpaceBeforeAuto = False
        .SpaceAfterAuto = False
        .FirstLineIndent = MillimetersToPoints(0)
        .CharacterUnitLeftIndent = 0
        .CharacterUnitFirstLineIndent = 0
    End With

End Sub


'tags für zeichenorientierte fv vergeben.
'
Private Sub tagZeichen(Vorlage, startTag, endTag)

Selection.HomeKey Unit:=wdStory
On Error Resume Next
With Selection.Find
    .ClearFormatting
    .Style = ActiveDocument.Styles(Vorlage)

    'filtern der ausgezeichneten absätze verhindert endlosschleifen.
    Do While .Execute(FindText:=Chr$(13), Forward:=True, Format:=True) _
             = True
        Selection.Style = ActiveDocument.Styles(_
                                        "Absatz-Standard-schriftart")
    Loop

    Selection.HomeKey Unit:=wdStory
    Do While .Execute(FindText:="", Forward:=True, Format:=True) = True
        If Selection.Text <> Chr$(13) Then
            Selection.MoveLeft
            Selection.Style = ActiveDocument.Styles(_
                                            "Absatz-Standardschriftart")
            Selection.TypeText Text:=startTag
            .Execute
            Selection.Style = ActiveDocument.Styles(_
                                            "Absatz-Standardschriftart")
```

```
            Selection.MoveRight
            Selection.Style = ActiveDocument.Styles(_
                                        "Absatz-Standardschriftart")
            Selection.TypeText Text:=endTag
        Else
        End If
    Loop
End With

End Sub
'tags für absatzorientierte fv vergeben.
'
Private Sub tagAbsatz(Vorlage, startTag, endTag)

Selection.HomeKey Unit:=wdStory
On Error Resume Next
   With Selection.Find
        .ClearFormatting
        .Style = ActiveDocument.Styles(Vorlage)
        Do While .Execute(FindText:="", Forward:=True, Format:=True) _
                 = True
            If Selection.End = ActiveDocument.Content.End Then
                If Selection.Text = Chr$(13) Then
                    Selection.MoveLeft
                    Selection.Style = ActiveDocument.Styles(_
                                      "Absatz-Standardschriftart")
                    Selection.TypeText Text:=startTag
                    Selection.Style = ActiveDocument.Styles(_
                                      "Absatz-Standardschriftart")
                    Selection.TypeText Text:=endTag
                Else
                    Selection.MoveLeft
                    Selection.Style = ActiveDocument.Styles(_
                                      "Absatz-Standardschriftart")
                    Selection.TypeText Text:=startTag
                    Selection.MoveDown Unit:=wdParagraph
                    Selection.Style = ActiveDocument.Styles(_
                                      "Absatz-Standardschriftart")
                    Selection.TypeText Text:=endTag
                End If
                GoTo Ende
            Else
```

```
                Selection.MoveLeft
                Selection.Style = ActiveDocument.Styles(_
                                  "Absatz-Standardschriftart")
                Selection.TypeText Text:=startTag
                Selection.MoveDown Unit:=wdParagraph
                Selection.MoveLeft
                Selection.Style = ActiveDocument.Styles(_
                                  "Absatz-Standardschriftart")
                Selection.TypeText Text:=endTag
            End If
        Loop
    End With

Ende:
End Sub
```

9.7 VERSUCHSPROTOKOLLE

Die Protokolle auf den folgenden Seiten Protokolle sind unkommentiert abgedruckt. Die Interpretation der Daten wird in Kapitel 6 und 7 vorgenommen.

9.7.1
Quark XPress 5.0 für Windows 2000
9.7.1.1
Copy & Paste

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	0	
absatzorientierte Formatvorlagen	0	
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	1	In feste Zeichen umgesetzt.
Fußnoten	1	In feste Zeichen umgesetzt.
– Fußnotenziffern	0	In feste Zeichen umgesetzt.
Hyperlinks	1	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Automatische Kapitelnummerierung	1	In feste Zeichen umgesetzt.
Automatische Aufzählungen	1	In feste Zeichen umgesetzt, Aufzählungszeichen korrekt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Absatz-getrennten Text umgewandelt.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	0	
Auszeichnungen per Hand		
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Werden sogar durch Leerzeichen simuliert.
Sonderzeichen		
Weicher Umbruch	0	In festen Umbruch umgewandelt.
Weiche Trennung	0	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	1	Summenzeichen fehlt (im ANSI-Encoding nicht enthalten).
Gesamter Text umgesetzt	1	Alle Fußnoten fehlen!
Gesamtwertung	9	

9.7.1.2
Drag & Drop

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	o	
absatzorientierte Formatvorlagen	o	
Verschlagwortung durch Indices	o	
Querverweise (auf Kapitel, Fußnoten)	o	
Fußnoten	o	
– Fußnotenziffern	o	
Hyperlinks	o	
platzierte Bilder	o	
Bildbeschriftung/-nummerierung	o	
Automatische Kapitelnummerierung	o	
Automatische Aufzählungen	o	
Tabelle mit Formatierung		
Umsetzung in Tabelle	o	
Zeichenformatierung	o	
Zellenformatierung	o	
Tabellenbeschriftung/-nummerierung	o	
Auszeichnungen per Hand		
Bold	o	
Kursiv	o	
Boldkursiv	o	
Falsche Kapitälchen	o	
Versalien	o	
Unterstreichung	o	
Durchstreichung	o	
Tiefergestellt	o	
Hochgestellt	o	
Andere Schrift	o	
Anderer Schriftgrad	o	
Linksbündig	o	
Blocksatz	o	
Rechtsbündig	o	
Zentriert	o	
Einzüge	o	
Sonderzeichen		
Weicher Umbruch	o	
Weiche Trennung	o	
Festes Divis	o	
Geschütztes Leerzeichen	o	
Seitenumbruch	o	
Gedankenstrich, Euro, Summenzeichen	o	
Gesamter Text umgesetzt	o	
Gesamtwertung	o	

Wird von XPress nicht unterstützt.

9.7.1.3
Word-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	2	Leider werden auch die nicht benutzten importiert. Bei Verwendung der Funktion »unbenutzte löschen« gerät XPress durcheinander und verändert die Stilvorlagenzuweisungen im Text. Also: Unbenutzte einzeln per Hand löschen!
absatzorientierte Formatvorlagen	2	Leider werden auch die nicht benutzten importiert. Bei Verwendung der Funktion »unbenutzte löschen« gerät XPress durcheinander und verändert die Stilvorlagenzuweisungen im Text. Also: Unbenutzte einzeln per Hand löschen!
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	1	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	0	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	0	Fehlt gänzlich!
Automatische Aufzählungen	0	Die Aufzählungszeichen fehlen.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Tabulator-getrennten Text umgewandelt.
Zeichenformatierung	2	Die Stilvorlagen werden übernommen.
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	1	Summenzeichen fehlt (im ANSI-Encoding nicht enthalten).
Gesamter Text umgesetzt	2	
Gesamtwertung	56	

9.7.1.4
RTF

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	o	
absatzorientierte Formatvorlagen	o	
Verschlagwortung durch Indices	o	
Querverweise (auf Kapitel, Fußnoten)	o	
Fußnoten	o	
– Fußnotenziffern	o	
Hyperlinks	o	
platzierte Bilder	o	
Bildbeschriftung/-nummerierung	o	
Automatische Kapitelnummerierung	o	
Automatische Aufzählungen	o	
Tabelle mit Formatierung		
Umsetzung in Tabelle	o	
Zeichenformatierung	o	
Zellenformatierung	o	
Tabellenbeschriftung/-nummerierung	o	
Auszeichnungen per Hand		
Bold	o	
Kursiv	o	
Boldkursiv	o	
Falsche Kapitälchen	o	
Versalien	o	
Unterstreichung	o	
Durchstreichung	o	
Tiefergestellt	o	
Hochgestellt	o	
Andere Schrift	o	
Anderer Schriftgrad	o	
Linksbündig	o	
Blocksatz	o	
Rechtsbündig	o	
Zentriert	o	
Einzüge	o	
Sonderzeichen		
Weicher Umbruch	o	
Weiche Trennung	o	
Festes Divis	o	
Geschütztes Leerzeichen	o	
Seitenumbruch	o	
Gedankenstrich, Euro, Summenzeichen	o	
Gesamter Text umgesetzt	o	
Gesamtwertung	o	

XPress konnte das RTF-File nicht importieren. Wahrscheinlich interpretiert der Filter nur ältere RTF-Versionen.

9.7.1.5
Nur-Text-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	0	
absatzorientierte Formatvorlagen	0	
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	1	In feste Zeichen umgesetzt.
Fußnoten	1	In feste Zeichen umgesetzt.
– Fußnotenziffern	0	In feste Zeichen umgesetzt.
Hyperlinks	1	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Automatische Kapitelnummerierung	1	In feste Zeichen umgesetzt.
Automatische Aufzählungen	1	In feste Zeichen umgesetzt, Aufzählungszeichen korrekt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Absatz-getrennten Text umgewandelt.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	0	
Auszeichnungen per Hand		
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Werden sogar durch Leerzeichen simuliert.
Sonderzeichen		
Weicher Umbruch	0	In festen Umbruch umgewandelt.
Weiche Trennung	0	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	1	Summenzeichen fehlt (im ANSI-Encoding nicht enthalten).
Gesamter Text umgesetzt	2	
Gesamtwertung	10	

Quark importiert lediglich das ANSI-Encoding. Unicode oder Macintosh-Standard-Encoding werden nicht unterstützt.

9.7.1.6
XML

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	o	
absatzorientierte Formatvorlagen	o	
Verschlagwortung durch Indices	o	
Querverweise (auf Kapitel, Fußnoten)	o	
Fußnoten	o	
– Fußnotenziffern	o	
Hyperlinks	o	
platzierte Bilder	o	
Bildbeschriftung/-nummerierung	o	
Automatische Kapitelnummerierung	o	
Automatische Aufzählungen	o	
Tabelle mit Formatierung		
Umsetzung in Tabelle	o	
Zeichenformatierung	o	
Zellenformatierung	o	
Tabellenbeschriftung/-nummerierung	o	
Auszeichnungen per Hand		
Bold	o	
Kursiv	o	
Boldkursiv	o	
Falsche Kapitälchen	o	
Versalien	o	
Unterstreichung	o	
Durchstreichung	o	
Tiefergestellt	o	
Hochgestellt	o	
Andere Schrift	o	
Anderer Schriftgrad	o	
Linksbündig	o	
Blocksatz	o	
Rechtsbündig	o	
Zentriert	o	
Einzüge	o	
Sonderzeichen		
Weicher Umbruch	o	
Weiche Trennung	o	
Festes Divis	o	
Geschütztes Leerzeichen	o	
Seitenumbruch	o	
Gedankenstrich, Euro, Summenzeichen	o	
Gesamter Text umgesetzt	o	
Gesamtwertung	o	

Die XML-Datei konnte aufgrund eines Lesefehlers nicht importiert werden:
Es erschien die Fehlermeldung, dass zu viele Elemente definiert seien oder das Dokument Endlosschleifen enthalte.

9.7.1.7
XPress-Marken

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
absatzorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Verschlagwortung durch Indices	0	Kann mit XPress-Marken nicht realisiert werden.
Querverweise (auf Kapitel, Fußnoten)	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Fußnoten	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
– Fußnotenziffern	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt. Im Fließtext korrekt mit Stilvorlagen ausgezeichnet.
Hyperlinks	0	Kann mit XPress-Marken nicht realisiert werden.
platzierte Bilder	0	Kann mit XPress-Marken nicht realisiert werden.
Bildbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die im Word zugewiesen wurde.
Automatische Kapitelnummerierung	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Automatische Aufzählungen	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	1	Tabulator getrennter Text. Kann mit XPress-Marken leider nicht umgesetzt werden.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Auszeichnungen per Hand		Die zusätzlichen Auszeichnungen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt (vom Nur-Text-Export aus Word).
Versalien	1	In feste Versalien umgesetzt (vom Nur-Text-Export aus Word).
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	
Sonderzeichen		Die Sonderzeichen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Weicher Umbruch	0	In festen Umbruch umgewandelt.
Weiche Trennung	0	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	1	Summenzeichen fehlt (im ANSI-Encoding nicht enthalten).
Gesamter Text umgesetzt	2	
Gesamtwertung	14	

9.7.2
InDesign 2.0 für Windows 2000
9.7.2.1
Copy & Paste

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Alle benutzten werden importiert. Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei bereits angelgten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
absatzorientierte Formatvorlagen	1	Alle benutzten werden importiert. Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei bereits angelgten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
Verschlagwortung durch Indices	2	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	2	
platzierte Bilder	2	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	2	In feste Zeichen umgesetzt.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	2	
Zeichenformatierung	2	Stilvorlagen werden übernommen.
Zellenformatierung	2	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	2	
Geschütztes Leerzeichen	2	
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	74	

InDesign führt einen RTF-Import durch.

9.7.2.2
Drag & Drop

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Alle benutzten werden importiert. Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei bereits angelgten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
absatzorientierte Formatvorlagen	1	Alle benutzten werden importiert. Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei bereits angelgten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
Verschlagwortung durch Indices	2	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	2	
platzierte Bilder	2	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	2	In feste Zeichen umgesetzt.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	2	
Zeichenformatierung	2	Stilvorlagen werden übernommen.
Zellenformatierung	2	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	2	
Geschütztes Leerzeichen	2	
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	74	

Wird das Datei-Symbol auf einen Textrahmen im InDesign-Fenster gezogen, führt InDesign einen Word-Import durch.

Drag & Drop von Fenster zu Fenster ergibt einen RTF-Import.

9.7.2.3
Word-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	2	Alle benutzten werden importiert. Bei bereits angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
absatzorientierte Formatvorlagen	1	Es fehlten die Stilvorlagen der Überschrifthierarchien 2 und 4. Dort fehlte auch die Kapitelnummerierung. Bei bereits angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
Verschlagwortung durch Indices	2	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	2	
platzierte Bilder	2	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	1	Es fehlten die Stilvorlagen der Überschrifthierarchien 2 und 4. Dort fehlte auch die Kapitelnummerierung.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	2	
Zeichenformatierung	2	Stilvorlagen werden übernommen.
Zellenformatierung	2	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	0	Nicht übernommen.
Geschütztes Leerzeichen	2	
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	72	

9.7.2.4
RTF

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Alle benutzten werden importiert. Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei bereits angelgten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
absatzorientierte Formatvorlagen	1	Alle benutzten werden importiert. Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei bereits angelgten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
Verschlagwortung durch Indices	2	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	2	
platzierte Bilder	2	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	2	In feste Zeichen umgesetzt.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	2	
Zeichenformatierung	2	Stilvorlagen werden übernommen.
Zellenformatierung	2	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	2	
Geschütztes Leerzeichen	2	
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	74	

9.7.2.5
Nur-Text-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	0	
absatzorientierte Formatvorlagen	0	
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	1	In feste Zeichen umgesetzt.
Fußnoten	1	In feste Zeichen umgesetzt.
– Fußnotenziffern	1	In feste Zeichen umgesetzt.
Hyperlinks	0	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Automatische Kapitelnummerierung	1	In feste Zeichen umgesetzt.
Automatische Aufzählungen	1	In feste Zeichen umgesetzt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Absatz-getrennten Text umgewandelt.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Auszeichnungen per Hand		
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Einzüge werden mit Leerzeichen simuliert (vom Word-Export-Filter).
Sonderzeichen		
Weicher Umbruch	0	
Weiche Trennung	0	
Festes Divis	0	Ergibt Importfehler!
Geschütztes Leerzeichen	2	
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	Im ANSI-Encoding fehlt das Summenzeichen. InDesign unterstützt aber Unicode und Mac-Encoding.
Gesamter Text umgesetzt	2	
Gesamtwertung	14	

InDesign unterstützt ANSI-, Mac- und Unicode-Encoding.

9.7.2.6
XML

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	2	
absatzorientierte Formatvorlagen	2	
Verschlagwortung durch Indices	1	Indices sind als Elementattribute integriert, können aber nicht der InDesign-Programmfunktion zugeordnet werden.
Querverweise (auf Kapitel, Fußnoten)	1	
Fußnoten	0	Fußnoten sind als Elementattribute integriert und stehen deshalb über das ganze Dokument verteilt.
– Fußnotenziffern	2	
Hyperlinks	1	Hyperlinks sind als Elementattribute integriert, können aber nicht der InDesign-Programmfunktion zugeordnet werden.
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	2	
Automatische Kapitelnummerierung	2	
Automatische Aufzählungen	2	
Tabelle mit Formatierung		
Umsetzung in Tabelle	1	Wurden korrekt ins XML-Tabellenmodell übertragen, können aber der InDesign-Funktion nicht zugeordnet werden.
Zeichenformatierung	2	
Zellenformatierung	1	Wurden korrekt ins XML-Tabellenmodell übertragen, können aber der InDesign-Funktion nicht zugeordnet werden.
Tabellenbeschriftung/-nummerierung	2	
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Die Codeeinrückung wurde unsinnigerweise komplett mit importiert.
Sonderzeichen		
Weicher Umbruch	0	
Weiche Trennung	0	
Festes Divis	0	
Geschütztes Leerzeichen	2	
Seitenumbruch	1	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	50	

9.7.2.7
Tagged Text

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
absatzorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Verschlagwortung durch Indices	0	Kann mit Programmieraufwand voll realisiert werden.
Querverweise (auf Kapitel, Fußnoten)	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Fußnoten	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
– Fußnotenziffern	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt. Im Fließtext korrekt mit Stilvorlagen ausgezeichnet.
Hyperlinks	0	Kann mit Programmieraufwand voll realisiert werden.
platzierte Bilder	0	Kann mit Tagged Text nicht realisiert werden.
Bildbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Automatische Kapitelnummerierung	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Automatische Aufzählungen	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	1	Tabulator getrennter Text. Kann mit Programmieraufwand voll realisiert werden.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Auszeichnungen per Hand		Die zusätzlichen Auszeichnungen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	
Sonderzeichen		Die Sonderzeichen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Weicher Umbruch	0	
Weiche Trennung	0	
Festes Divis	0	
Geschütztes Leerzeichen	2	
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	Das Summenzeichen ist im ANSI-Encoding nicht definiert, kann aber über ein anderes Encoding transportiert werden.
Gesamter Text umgesetzt	2	
Gesamtwertung	17	

9.7.3
Quark XPress 5.0 für MacOS 9.2.1

9.7.3.1
Copy & Paste

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	0	
absatzorientierte Formatvorlagen	0	
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	1	In feste Zeichen umgesetzt.
Fußnoten	1	In feste Zeichen umgesetzt.
– Fußnotenziffern	0	In feste Zeichen umgesetzt.
Hyperlinks	1	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Automatische Kapitelnummerierung	1	In feste Zeichen umgesetzt.
Automatische Aufzählungen	1	In feste Zeichen umgesetzt, Aufzählungszeichen korrekt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Absatz-getrennten Text umgewandelt.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	0	
Auszeichnungen per Hand		
Bold	1	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Werden sogar durch Leerzeichen simuliert.
Sonderzeichen		
Weicher Umbruch	0	In festen Umbruch umgewandelt.
Weiche Trennung	0	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	12	

XPress führt einen Nur-Text-Import lediglich im Systemencoding Macintosh-Standard durch. ANSI- und Unicode-Encoding werden nicht unterstützt.

9.7.3.2
Drag & Drop

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	o	
absatzorientierte Formatvorlagen	o	
Verschlagwortung durch Indices	o	
Querverweise (auf Kapitel, Fußnoten)	o	
Fußnoten	o	
– Fußnotenziffern	o	
Hyperlinks	o	
platzierte Bilder	o	
Bildbeschriftung/-nummerierung	o	
Automatische Kapitelnummerierung	o	
Automatische Aufzählungen	o	
Tabelle mit Formatierung		
Umsetzung in Tabelle	o	
Zeichenformatierung	o	
Zellenformatierung	o	
Tabellenbeschriftung/-nummerierung	o	
Auszeichnungen per Hand		
Bold	o	
Kursiv	o	
Boldkursiv	o	
Falsche Kapitälchen	o	
Versalien	o	
Unterstreichung	o	
Durchstreichung	o	
Tiefergestellt	o	
Hochgestellt	o	
Andere Schrift	o	
Anderer Schriftgrad	o	
Linksbündig	o	
Blocksatz	o	
Rechtsbündig	o	
Zentriert	o	
Einzüge	o	
Sonderzeichen		
Weicher Umbruch	o	
Weiche Trennung	o	
Festes Divis	o	
Geschütztes Leerzeichen	o	
Seitenumbruch	o	
Gedankenstrich, Euro, Summenzeichen	o	
Gesamter Text umgesetzt	o	
Gesamtwertung	o	

XPress unterstützt kein Drag & Drop.

9.7.3.3
Word-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	2	Leider werden auch die nicht benutzten importiert. Bei Verwendung der Funktion »unbenutzte löschen« gerät XPress durcheinander und verändert die Stilvorlagenzuweisungen im Text. Also: Unbenutzte einzeln per Hand löschen!
absatzorientierte Formatvorlagen	2	Leider werden auch die nicht benutzten importiert. Bei Verwendung der Funktion »unbenutzte löschen« gerät XPress durcheinander und verändert die Stilvorlagenzuweisungen im Text. Also: Unbenutzte einzeln per Hand löschen!
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	0	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	0	Fehlt gänzlich!
Automatische Aufzählungen	0	Die Aufzählungszeichen fehlen.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Tabulator-getrennten Text umgewandelt.
Zeichenformatierung	2	Die Stilvorlagen werden übernommen.
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	58	

9.7.3.4
RTF

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	o	
absatzorientierte Formatvorlagen	o	
Verschlagwortung durch Indices	o	
Querverweise (auf Kapitel, Fußnoten)	o	
Fußnoten	o	
– Fußnotenziffern	o	
Hyperlinks	o	
platzierte Bilder	o	
Bildbeschriftung/-nummerierung	o	
Automatische Kapitelnummerierung	o	
Automatische Aufzählungen	o	
Tabelle mit Formatierung		
Umsetzung in Tabelle	o	
Zeichenformatierung	o	
Zellenformatierung	o	
Tabellenbeschriftung/-nummerierung	o	
Auszeichnungen per Hand		
Bold	o	
Kursiv	o	
Boldkursiv	o	
Falsche Kapitälchen	o	
Versalien	o	
Unterstreichung	o	
Durchstreichung	o	
Tiefergestellt	o	
Hochgestellt	o	
Andere Schrift	o	
Anderer Schriftgrad	o	
Linksbündig	o	
Blocksatz	o	
Rechtsbündig	o	
Zentriert	o	
Einzüge	o	
Sonderzeichen		
Weicher Umbruch	o	
Weiche Trennung	o	
Festes Divis	o	
Geschütztes Leerzeichen	o	
Seitenumbruch	o	
Gedankenstrich, Euro, Summenzeichen	o	
Gesamter Text umgesetzt	o	
Gesamtwertung	o	

XPress stellt unter MacOS keinen RTF-Importfilter zur Verfügung.

9.7.3.5
Nur-Text-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	0	
absatzorientierte Formatvorlagen	0	
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	1	In feste Zeichen umgesetzt.
Fußnoten	1	In feste Zeichen umgesetzt.
– Fußnotenziffern	0	In feste Zeichen umgesetzt.
Hyperlinks	1	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Automatische Kapitelnummerierung	1	In feste Zeichen umgesetzt.
Automatische Aufzählungen	1	In feste Zeichen umgesetzt, Aufzählungszeichen korrekt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Absatz-getrennten Text umgewandelt.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	0	
Auszeichnungen per Hand		
Bold	1	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Werden sogar durch Leerzeichen simuliert.
Sonderzeichen		
Weicher Umbruch	0	In festen Umbruch umgewandelt.
Weiche Trennung	0	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	12	

XPress unterstützt nur das Systemencoding Macintosh-Standard.

9.7.3.6
XML

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	o	
absatzorientierte Formatvorlagen	o	
Verschlagwortung durch Indices	o	
Querverweise (auf Kapitel, Fußnoten)	o	
Fußnoten	o	
– Fußnotenziffern	o	
Hyperlinks	o	
platzierte Bilder	o	
Bildbeschriftung/-nummerierung	o	
Automatische Kapitelnummerierung	o	
Automatische Aufzählungen	o	
Tabelle mit Formatierung		
Umsetzung in Tabelle	o	
Zeichenformatierung	o	
Zellenformatierung	o	
Tabellenbeschriftung/-nummerierung	o	
Auszeichnungen per Hand		
Bold	o	
Kursiv	o	
Boldkursiv	o	
Falsche Kapitälchen	o	
Versalien	o	
Unterstreichung	o	
Durchstreichung	o	
Tiefergestellt	o	
Hochgestellt	o	
Andere Schrift	o	
Anderer Schriftgrad	o	
Linksbündig	o	
Blocksatz	o	
Rechtsbündig	o	
Zentriert	o	
Einzüge	o	
Sonderzeichen		
Weicher Umbruch	o	
Weiche Trennung	o	
Festes Divis	o	
Geschütztes Leerzeichen	o	
Seitenumbruch	o	
Gedankenstrich, Euro, Summenzeichen	o	
Gesamter Text umgesetzt	o	
Gesamtwertung	o	

Die XML-Datei konnte aufgrund eines Lesefehlers nicht importiert werden.

9.7.3.7
XPress-Marken

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
absatzorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Verschlagwortung durch Indices	0	Kann mit XPress-Marken nicht realisiert werden.
Querverweise (auf Kapitel, Fußnoten)	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Fußnoten	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
– Fußnotenziffern	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt. Im Fließtext korrekt mit Stilvorlagen ausgezeichnet.
Hyperlinks	0	Kann mit XPress-Marken nicht realisiert werden.
platzierte Bilder	0	Kann mit XPress-Marken nicht realisiert werden.
Bildbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die im Word zugewiesen wurde.
Automatische Kapitelnummerierung	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Automatische Aufzählungen	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	1	Tabulator getrennter Text. Kann mit XPress-Marken leider nicht umgesetzt werden.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Auszeichnungen per Hand		Die zusätzlichen Auszeichnungen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt (vom Nur-Text-Export aus Word).
Versalien	1	In feste Versalien umgesetzt (vom Nur-Text-Export aus Word).
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	
Sonderzeichen		Die Sonderzeichen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Weicher Umbruch	0	In festen Umbruch umgewandelt.
Weiche Trennung	0	
Festes Divis	0	In Einzug-hier-Zeichen umgewandelt.
Geschütztes Leerzeichen	0	In Viertelgeviert-Zeichen umgewandelt.
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	15	

9.7.4
InDesign 2.0 für MacOS 9.2.1
9.7.4.1
Copy & Paste

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	0	Der gesamte Text wird als Arial BoldItalic formatiert.
absatzorientierte Formatvorlagen	0	
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	0	Es wurden keine Fußnoten importiert.
– Fußnotenziffern	1	In feste Zeichen umgesetzt.
Hyperlinks	0	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Automatische Kapitelnummerierung	2	In feste Zeichen umgesetzt.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	1	In Tabulator-getrennten Text umgewandelt.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Auszeichnungen per Hand		
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	
Sonderzeichen		
Weicher Umbruch	0	In festen Umbruch umgewandelt.
Weiche Trennung	0	In feste Trennung umgewandelt.
Festes Divis	0	
Geschütztes Leerzeichen	0	
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	0	Es wurden keine Fußnoten importiert!
Gesamtwertung	17	

9.7.4.2
Drag & Drop

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Einige Formatierungen werden falsch importiert (kursiv statt normal). Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
absatzorientierte Formatvorlagen	1	Einige Formatierungen werden falsch importiert (kursiv statt normal). Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
Verschlagwortung durch Indices	2	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	2	
platzierte Bilder	2	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	2	In feste Zeichen umgesetzt.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	2	
Zeichenformatierung	2	Stilvorlagen werden übernommen.
Zellenformatierung	2	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	2	
Geschütztes Leerzeichen	2	
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	74	

Wird das Datei-Symbol auf einen Textrahmen im InDesign-Fenster gezogen, führt InDesign einen Word-Import durch.

Drag & Drop von Fenster zu Fenster ergibt einen RTF-Import.

9.7.4.3
Word-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Einige Formatierungen werden falsch importiert (kursiv statt normal). Bei bereits angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
absatzorientierte Formatvorlagen	1	Einige Formatierungen werden falsch importiert (kursiv statt normal). Es fehlen die Stilvorlagen der Überschriften 2 und 4, wo auch die Kapitelnummerierung fehlte. Bei angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
Verschlagwortung durch Indices	2	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	2	
platzierte Bilder	2	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	1	Es fehlten die Stilvorlagen der Überschriften 2 und 4. Dort fehlte auch die Kapitelnummerierung.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	2	
Zeichenformatierung	2	Stilvorlagen werden übernommen.
Zellenformatierung	2	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	1	In normales Divis umgesetzt.
Geschütztes Leerzeichen	2	
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	72	

9.7.4.4
RTF

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Einige Formatierungen werden falsch importiert (kursiv statt normal). Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
absatzorientierte Formatvorlagen	1	Einige Formatierungen werden falsch importiert (kursiv statt normal). Leider exportiert Word seine vordefinierten Formatvorlagen in englischer Sprache ins RTF. Bei angelegten Stilvorlagen erscheint ein »+« in der Stilvorlagenpalette.
Verschlagwortung durch Indices	2	
Querverweise (auf Kapitel, Fußnoten)	2	In feste Zeichen umgesetzt.
Fußnoten	2	In feste Zeichen mit Stilvorlagen umgesetzt.
– Fußnotenziffern	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Hyperlinks	2	
platzierte Bilder	2	
Bildbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Automatische Kapitelnummerierung	2	In feste Zeichen umgesetzt.
Automatische Aufzählungen	2	Aufzählungszeichen korrekt, mit Tabulator vom Text getrennt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	2	
Zeichenformatierung	2	Stilvorlagen werden übernommen.
Zellenformatierung	2	
Tabellenbeschriftung/-nummerierung	2	In feste Zeichen mit Stilvorlagen umgesetzt.
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	2	
Blocksatz	2	
Rechtsbündig	2	
Zentriert	2	
Einzüge	2	
Sonderzeichen		
Weicher Umbruch	2	
Weiche Trennung	2	
Festes Divis	2	
Geschütztes Leerzeichen	2	
Seitenumbruch	2	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	74	

9.7.4.5
Nur-Text-Format

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	0	
absatzorientierte Formatvorlagen	0	
Verschlagwortung durch Indices	0	
Querverweise (auf Kapitel, Fußnoten)	1	In feste Zeichen umgesetzt.
Fußnoten	1	In feste Zeichen umgesetzt.
– Fußnotenziffern	1	In feste Zeichen umgesetzt.
Hyperlinks	0	
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Automatische Kapitelnummerierung	1	In feste Zeichen umgesetzt.
Automatische Aufzählungen	1	In feste Zeichen umgesetzt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	0	In Absatz-getrennten Text umgewandelt.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	1	In feste Zeichen umgesetzt.
Auszeichnungen per Hand		
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Einzüge werden mit Leerzeichen simuliert (vom Word-Export-Filter).
Sonderzeichen		
Weicher Umbruch	0	
Weiche Trennung	0	
Festes Divis	1	In normales Divis umgesetzt.
Geschütztes Leerzeichen	2	
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	15	

InDesign unterstützt ANSI-, Mac- und Unicode-Encoding.

9.7.4.6
XML

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	2	
absatzorientierte Formatvorlagen	2	
Verschlagwortung durch Indices	1	Indices sind als Elementattribute erhalten, können aber nicht der InDesign-Programmfunktion zugeordnet werden.
Querverweise (auf Kapitel, Fußnoten)	1	
Fußnoten	0	Fußnoten sind ials Elementattribute erhalten, und stehen somit quer über den Text verteilt.
– Fußnotenziffern	2	
Hyperlinks	1	Hyperlinks sind als Elementattribute erhalten, können aber nicht der InDesign-Programmfunktion zugeordnet werden.
platzierte Bilder	0	
Bildbeschriftung/-nummerierung	2	
Automatische Kapitelnummerierung	2	
Automatische Aufzählungen	2	
Tabelle mit Formatierung		
Umsetzung in Tabelle	1	Wurden korrekt ins XML-Tabellenmodell übertragen, können aber der InDesign-Funktion nicht zugeordnet werden.
Zeichenformatierung	2	
Zellenformatierung	1	Wurden korrekt ins XML-Tabellenmodell übertragen, können aber der InDesign-Funktion nicht zugeordnet werden.
Tabellenbeschriftung/-nummerierung	2	
Auszeichnungen per Hand		
Bold	2	
Kursiv	2	
Boldkursiv	2	
Falsche Kapitälchen	2	
Versalien	2	
Unterstreichung	2	
Durchstreichung	2	
Tiefergestellt	2	
Hochgestellt	2	
Andere Schrift	2	
Anderer Schriftgrad	2	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	Die Codeeinrückung wurde unsinnigerweise komplett mit importiert.
Sonderzeichen		
Weicher Umbruch	0	
Weiche Trennung	0	
Festes Divis	0	
Geschütztes Leerzeichen	2	
Seitenumbruch	1	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	50	

9.7.4.7
Tagged Text

Merkmal	Wert	Bemerkung
zeichenorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
absatzorientierte Formatvorlagen	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Verschlagwortung durch Indices	0	Kann mit Programmieraufwand voll realisiert werden.
Querverweise (auf Kapitel, Fußnoten)	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Fußnoten	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
– Fußnotenziffern	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt. Im Fließtext korrekt mit Stilvorlagen ausgezeichnet.
Hyperlinks	0	Kann mit Programmieraufwand voll realisiert werden.
platzierte Bilder	0	Kann mit Tagged Text nicht realisiert werden.
Bildbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Automatische Kapitelnummerierung	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Automatische Aufzählungen	1	Alle umgesetzt wie es der Nur-Text-Export von Word vorgibt.
Tabelle mit Formatierung		
Umsetzung in Tabelle	1	Tabulator getrennter Text. Kann mit Programmieraufwand voll realisiert werden.
Zeichenformatierung	0	
Zellenformatierung	0	
Tabellenbeschriftung/-nummerierung	1	Stilvorlagen werden korrekt übernommen, allerdings ohne Formatierung, die in Word zugewiesen wurde.
Auszeichnungen per Hand		Die zusätzlichen Auszeichnungen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Bold	0	
Kursiv	0	
Boldkursiv	0	
Falsche Kapitälchen	0	In feste Versalien umgesetzt.
Versalien	1	In feste Versalien umgesetzt.
Unterstreichung	0	
Durchstreichung	0	
Tiefergestellt	0	
Hochgestellt	0	
Andere Schrift	0	
Anderer Schriftgrad	0	
Linksbündig	0	
Blocksatz	0	
Rechtsbündig	0	
Zentriert	0	
Einzüge	0	
Sonderzeichen		Die Sonderzeichen wurden im Beispielmakro nicht behandelt, sind aber alle realisierbar.
Weicher Umbruch	0	
Weiche Trennung	0	
Festes Divis	0	
Geschütztes Leerzeichen	2	
Seitenumbruch	0	
Gedankenstrich, Euro, Summenzeichen	2	
Gesamter Text umgesetzt	2	
Gesamtwertung	17	

9·7·5
Punktevergleich

Test	Windows		Macintosh	
	XPress	InDesign	XPress	InDesign
Copy & Paste	10[1]	74[2]	12[1]	17
Drag & Drop	0[–]	74[2]	0[–]	74[2]
Wordfilter	56	72	58	72
RTF-Filter	0[F]	74	0[–]	74
Nur-Text-Filter	10	14	12	15
XML-Import	0[F]	50	0[F]	50
Steuercodes	14[3]	17[3]	15[3]	17[3]
Gesamt	90	375	97	319
[max. (je 76)	532	532	532	532]

F Fehler
– nicht implementiert
1 Es wurde ein vergleichbarer Nur-Text-Import durchgeführt.
2 Es wurde ein vergleichbarer RTF-Import durchgeführt.
3 Die Punktezahl lässt sich durch entsprechende Programmierung noch erheblich steigern, so dass sie sich dem Vergleich entzieht.

9.8 QUELLENVERZEICHNIS

Das Internet hat sich zum gigantischen Informationspool entwickelt. Es beherbergt eine Fülle an Quellen mit hohem Informationsgehalt, auch wenn diese nicht immer leicht zu finden sind. Leider liegen wertvolle Informationen allzu nah an nichtigen, so dass umfangreiche Recherche nötig ist, um die Spreu vom Weizen zu trennen. Der Lohn jedoch ist hohe Aktualität und das Schauen aus vielen verschiedenen Blickwinkeln auf ein Thema, da jeder Autor andere Interessen mit seinem Artikel verfolgt. Deshalb wurde diese Arbeit hauptsächlich aus Non-Print-Quellen erstellt.

9.8.1
Print-Quellen

[1] Adobe Systems: Adobe, Produkt-Palette. Stand 03/02.

[2] dtv-Lexikon: Band 4. Deutscher Taschenbuch Verlag, München, 1999.

[3] dtv-Lexikon: Band 11. Deutscher Taschenbuch Verlag, München, 1999.

[4] DUDEN: Band 5, Fremdwörterbuch. Bibliographisches Institut, Mannheim, 4. Auflage, 1982.

[5] Goik, Martin: Skript zur Vorlesung »Dokumenterstellung«. WS 1999/2000, Fachhochschule Stuttgart – Hochschule der Medien.

[6] Heuer, Steffan: Was Wirtschaft treibt – die Erfinder von Word, Excel und Powerpoint. BrandEins, 3.2002.

[7] Ott, Tobias: Skript zur Vorlesung »Elektronisches Publizieren«. WS 1999/2000, Fachhochschule Stuttgart – Hochschule der Medien.

[8] Siemoneit, Manfred/Zeitvogel, Wolfgang: Satzherstellung. Vom Bleisatz zum Computer Publishing. Polygraph Verlag, 3. Auflage, 1992.

[9] Microsoft Corporation: Microsoft Word 2000 Visual Basic Sprachverzeichnis. Microsoft Press, 2000.

[10] Halvorsen, Michael/Kinata, Chris: Microsoft Word 97 Visual Basic. Microsoft Press, 1998.

9.8.2
Non-Print-Quellen

[11] Adobe Systems Inc.: Adobe InDesign Programming Guide. 1997–2000, Bestandteil des InDesign SDK, online unter: http://partners.adobe.com/asn/developer/indesign/sdk.html, Stand: 19. 9. 2002.

[12] Adobe Systems Inc.: Homepage des PostScript-Standards. Online unter: http://www.adobe.com/products/postscript/main.html (Stand: 19. 9. 2002).

[13] Alvestrand, H.: IETF Policy on Character Sets and Languages, RFC 2277. 1998, online unter: ftp://ftp.isi.edu/in-notes/rfc2277.txt (Stand: 1. 4. 2002).

[14] Apple Computer: Inside Macintosh: Aqua Human Interface Guidelines. 2001, online unter: http://developer.apple.com/techpubs/macosx/Essentials/AquaHIGuidelines/index.html (Stand: 8. 4. 2002).

[15] Apple Computer: Macintosh/PC Compatibility Guide. 2002, online unter: http://www.apple.com/uk/smallbusiness/mac_pc/ (Stand: 13. 4. 2002).

[16] Becker, David: Corel zielt auf unzufriedene Microsoft-Kunden. ZDNet, 12.8.2002, online unter: http://techupdate.zdnet.de/story/0,,t423-s2120685-p1,00.html, Stand: 11. 9. 2002.

[17] Bisguier, Arthur: Ten Tips to Winning Chess – by International Grandmaster Arthur Bisguier. 1998, online unter: http://www.uschess.org/beginners/ten/index.html (Stand: 5. 7. 2002).

[18] Bricknell, K. J.: Chapter 18: Scrap. 2000, online unter: http://www.mactech.com/macintosh-c/classic-chap18-1.html (Stand 26. 3. 2002).

[19] Brockhaus Multimedial 2001, CD-ROM. Bibliographisches Institut & F. A. Brockhaus AG, 2001.

[20] Czyborra, Roman: The ISO 8859 Alphabet Soup. 1998, online unter: http://czyborra.com/charsets/iso8859.html (Stand: 13. 4. 2002).

[21] Czyborra, Roman: Unicode Transformation Formats: UTF-8 & Co. 1998, online unter: http://czyborra.com/utf/index.html (Stand: 13. 4. 2002).

[22] Deutsche Bahn AG: Online-Reiseauskunft. Online unter: http://reiseauskunft.bahn.de/ (Stand: 19. 4. 2002).

[23] EMC² Corp.: EMC E-Infostructure. Online unter: http://www.emc.com/products/e-info/index.jsp (Stand: 10. 9. 2002).
[24] Jones International and Jones Digital Century: Quark Inc. 1994–99, online unter: http://www.digitalcentury.com/encyclo/update/quark.html, Stand: 19. 9. 2002.
[25] Kleyer, Christian/Oyen, Daniel/Reuse, Svend: SGML – Grundkonzeption und Zusammenhang mit XML. 2001, online unter: http://th-o.de/sgml/sgml_xml_konzept.pdf (Stand: 11. 9. 2002).
[26] Korpela, Jukka: A tutorial on character code issues. 2001, online unter: http://www.cs.tut.fi/~jkorpela/chars.html (Stand: 13. 4. 2002).
[27] Lessard, Daniel: Windows. Online unter: http://members.fortunecity.com/pcmuseum/windows.htm (Stand: 19. 6. 2002).
[28] Macherius, Ingo: XML: Professionelle Alternative zu HTML – Revolution der Experten. 1996, online unter: http://www.heise.de/ix/artikel/1997/06/106/ (Stand: 14. 9. 2002).
[29] Mesa, Andy F.: A History of the Graphical User Interface. 1998, online unter: http://applemuseum.bott.org/sections/gui.html (Stand: 19. 6. 2002).
[30] Mesa, Andy F.: Apple History Timeline. 1998, online unter: http://applemuseum.bott.org/sections/history.html (Stand: 19. 6. 2002).
[31] Otterstetter, Gabriele: Dritte Welt und der Aufbruch in das Informationszeitalter am Beispiel Internet. 1998, Berlin, online unter: http://www.tu-berlin.de/fb2/as3/as3w/dipl_ott/dip_inh.htm (Stand: 2. 10. 2001).
[32] Quark Inc.: History. 2002, online unter: http://www.quark.com/about/profile/history.html, (Stand: 19. 9. 2002).
[33] Shane: Where is Quark for OS X? 16. 9. 2002, online unter: http://www.geek.com/news/geeknews/2002Sep/bma20020916016358.htm (Stand: 19. 9. 2002).
[34] Simonsen, Keld und andere: Character Sets. 2002, online unter: http://www.iana.org/assignments/character-sets (Stand: 13. 4. 2002).
[35] Simonsen, Keld: Character Mnemonics & Character Sets. 1992, online unter: ftp://ftp.isi.edu/in-notes/rfc1345.txt (Stand: 13. 4. 2002).
[36] Trotot, Jean Christophe: History of Apple Macintosh Operating System. Online unter: http://perso.club-internet.fr/jctrotot/Perso/History.html (Stand: 19. 6. 2002).
[37] http://www.wissen.de (Stand: 9. 9. 2002).

9.9 ABKÜRZUNGS- UND DEFINITIONSVERZEICHNIS

AFP
Apple File Protocol

AGP
Accelerated Graphics Port

ANSI
American Nation Standards Institute

API
Aplication Programing Interface – Schnittstellendefinitionen, die vom Betriebssystem- oder Applikationshersteller zur Verfügung gestellt werden. Mit ihnen kann ein Programmierer System- oder Anwendungsfunktionen direkt ansprechen und für seine Zwecke nutzen.

Application Heap
Speicherbereich, der einer Applikation vom Betriebssystem zur Verfügung gestellt wird.

ASCII
American Standard Code for Information Interchange

ATSUI
Apple Type Services for Unicode Imaging

AVI
Audio Video Interlaced

BIOS
Basic Input/Output System

BMP
Bitmap Picture

BNF
Backus-Naur-Form

CAD
Computed Aided Design

CAP
Computer Aided Publishing

CCS
Coded Character Set

CDFS
CD-ROM File System

CD-ROM
Compact Disk Read Only Memory

CEF
Character Encoding Form

CES
Character Encoding Scheme

Character Encoding Form
Zuordnung des CCS zu einer Binärzahl (Code Unit).

Character Encoding Scheme
Regeln, wie CEF auf Byte-Blöcke verteilt werden.

Character Repertoire
Zeichenvorrat eines Codes.

Charset
Kombination CCS, CES.

Code Number
Eindeutige, nicht-negative Ganzzahl, die die Position eines Zeichens in einem Codesystem festlegt.

Code Unit
Binärzahl der Code Number.

Coded Character Set
Zuordnungsregelwerk Code Number zu Character Repertoire.

CP
Codepage

CPU
Central Processing Unit

CR
Carriage Return

CSS
Cascading Stylesheets

DCS
Document Color Separation

DEC
Dezimal

DLL
Dynamic Link Library

DML
Data Manipulation Language

DSSSL
Document Style Semantics and Specification Language

DTD
Document Type Definition

DTP
Desktop Publishing

DV
Digital Video

DVD
Digital Video Disc

DXF
Drawing eXchange Format

EBCDIC
Extended Binary Coded Interchange Code

EDV
Elektronische Datenverarbeitung

EFS
Encrypting File System

EPS
Encapsulated PostScript

Event
Ein Event (Ereignis) entsteht im Betriebssystem, wenn ein Eingabegerät Signale sendet (Mausbewegung, Klicken), ein Programm eine Datenbearbeitung beendet hat oder ähnliches.

FAT
File Allocation Table

GDI
Graphical Device Interface

GIF
Graphic Interchange Format

Glyphe
Zeichnerische, visuelle Erscheinungsform eines Zeichens.

GUI
Graphical User Interface

HAL
Hardware Abstraction Layer

HEX
Hexadezimal

HFS
Hierachical File System

HTML
Hypertext Markup Language

HTTP
Hypertext Transfer Protocol

IANA
Internet Assigned Numbers Authority

I/O
Input/Output

IIS
Internet Information Server

Instanz
Ein XML/SGML-Dokument.

ISO
International Standard Organisation

IT
Information Technology

IWT
Immer wiederkehrender Text

J2SE
Java 2 Standard Edition

JPEG
Joint Picture Expert Group

Kernel
Schnittstelle zwischen Systemsoftware und Mikroprozessor (CPU). Er wandelt Befehle in Prozessor-Maschinencode.

LAN
Local Area Network

LF
Line Feed

LPC
Local Procedure Call

LUT
Look-up Table

Metainformationen
Informationen über Informationen.

MIME
Multipurpose Internet Mail Extensions

M-JPEG
Motion-Joint Picture Expert Group

MPEG
motion pictures expert group

MRJ
MacOS Runtime for Java

MS
Microsoft

Multiprozessing
Ermöglicht Applikationen die Nutung mehrerer Prozessoren in einem System.

Multitasking
Teilt die zur Verfügung stehende Prozessorzeit so auf, dass meherere Applikationen quasi gleichzeitig auf einem Prozessor laufen und arbeiten können.

NFS
Network File System

NKE
Network Kernel Extension

NTFS
New Technology File System

OLE
Object Linking and Embedding

OpenGL
Open Graphics Language

OS
Operating System

PC
Personal Computer

PDF
Portable Document Format

PERL
Practical Extraction and Reporting Language

PHP
PHP: Hypertext Preprocessor

PICT
Picture

PIF
Program Information File

PNG
Portable Network Format

PPC
PowerPC

Prozess
Prozess ist ein in der Ausführung bedindliches Programm.

RAID
Redundant Array of Independant Discs

RAM
Random Access Memory

RFC
Request for Comments

ROM
Read Only Memory

RTF
Rich Text Format

RTP
Real Time Protocol

RTSP
Real Time Streaming Protocol

SDK
Software Developer Kit – Enthält die wichtigen programmspezifischen APIs, die zur Programmierung von Zusatzprogrammen für das Softwareprodukt notwendig sind.

SGML
Standard Generalized Markup Language

SMP
Symetrical Multiprocessing

SQL
Structured Query Language

Subset
Untermenge von Zeichen in einem umfassenderen Character Repertoire.

SVG
Scaleable Vector Format

TAGC
Tag Close

TAGO
Tag Open

TCP
Transmission Control Protocol

TCP/IP
Transmission Protocol/Internet Protocol

TEC
Text Encoding Converter

Thread
Threads sind unabhängige kleine Subprogramme, die im Speicher verteilt laufen und von ihrem Mutterprogramm genutzt werden.

TIFF
Tagged Image File-Format

UCS
Universal Character Set

UDF
Universal Disc Format

UFS
Universal File System

UI
User Interface

URL
Uniform Resource Locator

USASCII
United States of America Standard Code for Information Interchange

USB
Universal Serial Bus

UTF
Universal Text Format

VBA
Visual Basic for Applications

VDM
Virtual DOS Machine

VFS
Virtual File System

VMM
Virtual Memory Manager

Windows NT
Windows New Technology

Window-Server
Fenstermanager – ist für die Erzeugung und Verwaltung der Fenster und die Erledigung von Teilaufgaben der Druckprozesse im Betriebssystem zuständig.

WML
Wireless Markup Language

WYSIWYG
What You See Is What You Get

XCHAR
eXtended Character Set

XML
eXtensible Markup Language

XSL
eXtensible Stylesheet Language

XSLT
eXtensible Stylesheet Language Transformation

Dank
Herzlich danke ich Antje Liesenfeld, Ulli Neutzling und Ralf Schnarrenberger, meinen kritischen Korrektoren, für die Opferung ihrer freien Zeit.

Ferner möchte ich all jenen danken, die mich bei der Entstehung dieser Arbeit unterstützt haben, und jenen, die so manches Mal privat wie beruflich auf mich verzichten mussten.

Impressum
Diese Arbeit wurde auf Tetenal DuoPrint, 130 g/m², mit einem GCC Elite XL 20/1200 und Epson Stylus 3000 in den alten Räumlichkeiten der Groothuis, Lohfert, Consorten – Gesellschaft für Formfindung und Sinneswandel mbH hergestellt. Die Produktion nahm der Autor komplett selbst vor.

Als Schrift kam die DTL Caspari$_{ST}$ 9,5/13,5 pt mit einer Laufweitenkorrektur von +2 Einheiten in Quark XPress 4.11 zum Einsatz. Gesetzt wurde auf einem Apple PowerBook G3.

```
Public Sub tagNow()
'zeigt dialog, um seq. datei zu laden, die tags enthält.
'
Dim DatName As String
Dim Button As Integer
With Application.Dialogs(wdDialogFileOpen)
    Button = .Display
    DateiName = .Name
End With
If Button = 0 Then End 'wenn abbrechen gedrückt: ende.

'speichert die anzahl der zeilen der seq. datei in der variablen n.
'
Open DateiName For Input As #1
n = 0
Do While Not EOF(1)  'loop until end of file.
    Line Input #1, TextLine
    n = n + 1 'zählt durchläufe mit.
Loop
Close #1

'dimensionierung des formatvorlagen-arrays (fv-arrays) und
'definition einiger konstanten:
'conName   = array-postition des fv-namens,
'conStart = array-position des start-tags,
'conEnd    = array-position des ende-tags,
'conArt    = array-position der absatz-/zeichenorientiert-info.
ReDim FVarray(n, 3) As String
Const conName As Integer = 0, conStart As Integer = 1, conEnd As Integer = 2, conArt

'auslesen der fv und tags aus der seq. datei und füllen des fv-arrays.
'
Open DateiName For Input As #1
t = 0
Do While Not EOF(1) 'loop until end of file.
    Line Input #1, TextLine
    'dann trennen nach delimiter <tab> und dem array zuweisen.
    LetzterTab = Len(TextLine) - 1
    myHelp = 0
    For h = 3 To 0 Step -1
        myHelp = LetzterTab
        Do While Mid(TextLine, LetzterTab, 1) <> Chr$(9)
            LetzterTab = LetzterTab - 1
            If LetzterTab = 0 Then Exit Do
        Loop
        If h = 3 Then
            FVarray(t, h) = Mid(TextLine, LetzterTab + 1, myHelp - LetzterTab +
        Else
            FVarray(t, h) = Mid(TextLine, LetzterTab + 1, myHelp - LetzterTab)
        End If
        LetzterTab = LetzterTab - 1
    Next h
```